MANEUVER DESIGN FOR FAST SATELLITE CIRCUMNAVIGATION

THESIS

Stanley D. Straight, Captain, USAF

AFIT/GA/ENY/04-M05

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GA/ENY/04-M05

MANEUVER DESIGN FOR FAST SATELLITE CIRCUMNAVIGATION

THESIS

Presented to the Faculty

Department of Aeronautics and Astronautics

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Astronautical Engineering

Stanley D. Straight, BS

Captain, USAF

March 2004

AFIT/GA/ENY/04-M05

MANEUVER DESIGN FOR FAST SATELLITE CIRCUMNAVIGATION

Stanley D. Straight, BS
Captain, USAF

Approved:

_____//Signed//_____	_____
Steven G. Tragesser (Chairman)	date
_____//Signed//_____	_____
William E. Wiesel (Member)	date
_____//Signed//_____	_____
Robert A. Howard (Member)	date

Abstract

The feasibility of satellite operations in close proximity to a reference satellite is of interest for both civilian and military applications. One such operation is circular circumnavigation in a time period less than the orbital period of the reference satellite. This thesis investigates a guidance scheme for such maneuvers involving impulsive burns at specific points within a specified toroidal region centered on the circular-orbiting reference satellite. Two analytical methods for determining the magnitude and direction of the impulses are demonstrated. These methods are then used as initial estimates in an optimization scheme to produce the minimum total required impulse.

To my wife, daughter, and son.

Acknowledgements

I would like to express my sincere appreciation to my advisor, Dr. Steven Tragesser, for his support and insightful guidance; his dedication to his students is exemplary. I would, also, like to thank my sponsor, Dr. T. Alan Lovell from the Space Vehicles Directorate, Air Force Research Laboratory, as well as Mr. Robert Howard and Dr. Frank Chavez for their support and considerable effort in review of my research.

A special thanks is extended to my fellow compatriot, Mr. Matthew Press, for his patience and understanding as a sounding board for all the ideas that didn't make it into this thesis and most that did.

Stanley D. Straight

Table of Contents

Page

Page

List of Figures

Figure Page

Figure Page

List of Tables

Table Page

MANEUVER DESIGN FOR FAST SATELLITE CIRCUMNAVIGATION

I. Introduction

Operations between two spacecraft in orbit have become of increasing interest to both the civilian and military communities. Satellite-to-satellite operations have been demonstrated since the beginning of human space endeavors. The most common relative satellite-to-satellite (relative motion) operations have been rendezvous between two cooperating spacecraft, but other proximity operations are becoming more important. In recent times, there has been considerable interest in orbiting satellites in close relative formations.

Considerable work has been done in the area of satellite formation flying: the design of formations (11; 13), their reconfiguration and maintenance (5), and formation evolution through orbital perturbations (3; 12; 14; 15). Portions of this work have focused on natural motion formations, establishing a relative position and velocity with respect to a reference orbit, and allowing the natural dynamics to produce elliptical motion in the relative frame. Reconfiguration of formations, another topic of study, has focused on optimizing propellant expenditure from one formation to another without necessarily focusing on the shape or time variation of the flight path. (5; 13)

Other proximity operations maneuvers are becoming more important in planning for such missions as on-orbit repair and refueling as well as potential damage inspection or identification of Resident Space Objects (RSO). (7; 10) Circumnavigating a chief

satellite with a deputy satellite provides the ability to inspect the chief from a variety of viewpoints. Often these viewpoints are required to be at a constant distance from the chief and therefore requiring circular circumnavigations. A circumnavigation is defined as the deputy's flight about a desired circular path (nominal path) with a specified orientation about a chief spacecraft.

Previous Work

The minimum propellant required for a circular circumnavigation is the natural motion circular formation (11:7-8), which requires the initial conditions to be set up in a very specific manner. These natural motion circumnavigations all have a circumnavigation period (rotating around the chief satellite through 360º) equal to the period of the chief. This period is on the order of 90 minutes for Low Earth Orbiting satellites with an altitude around 400 km, and increases as the altitude increases.

Circumnavigation times less than the chief's orbital period, are termed 'fast'. These fast circumnavigation maneuvers have utility in the operation of a proposed 'inspector' micro-satellite (4:1). In order to determine time changing phenomena on the chief spacecraft, these maneuvers need to be accomplished less than the orbital period of the chief satellite (4:1). As the required time for a circumnavigation decreases, the total impulse required increases to perform the circumnavigation. Minimizing the total impulse for a given maneuver allows for greater operational flexibility, increased sorties for a given amount of propellant, and potentially increase the lifetime for any given satellite performing these maneuvers.

The theoretical impulse requirements for differing circumnavigation times and orientations of a nominal circular path have been demonstrated (4: 1-2). Their method assumed continuous control to produce perfect circular motion with respect to the chief. After simplifications, the unperturbed Hill's equations are derived as (4:2; 14:377)

$$\begin{aligned} \ddot{x} - 2n\dot{y} - 3n^2 x &= f_x \\ \ddot{y} + 2n\dot{x} &= f_y \\ \ddot{z} + n^2 z &= f_z \end{aligned} \tag{1}$$

where x, y, and z represent the position as a function of time relative to the chief, and "f_x, f_y, and f_z are the propulsive forces per unit mass" (4:2). From these equations the total impulse, required to follow a path defined using Equation 1 is represented as (4:2):

$$\Delta V = \int_0^T \sqrt{f_x^2 + f_y^2 + f_z^2}\, dt \tag{2}$$

where T is the time required to follow the total path. This theoretical impulse is informative in understanding the total forces required to follow a specific path, but often the path is only constrained to lie within a certain volume. Exact adherence to a defined circumnavigation path is not always necessary; operationally, it is conjectured only a very small percentage of the total flight path is required to be at a certain position and time relative to the chief. The rest of the flight path is then constrained to be distance away from nominal path, or in some instances a minimum distance to the chief spacecraft/object.

Problem Statement

For the problem investigated here, the deputy will perform the circumnavigation through a set of discrete impulses in a specified time of flight (TOF). The deputy's flight path must perform a full 2π angular rotation about the chief without doubling back on itself. The placement and number of each burn point must be determined as well as the required direction and magnitude of each individual impulse.

The deputy is also constrained to stay within a specified distance from the desired or 'nominal' path during the circumnavigation maneuver. This constraint allows for operational considerations such as collision avoidance and operational requirements for the deputy's payload. The payload is postulated to potentially be a remote sensing detector (visual, infrared, etc…) where the distance to the chief can become an important operational factor.

There are two probable cases of general rules on constraining the placement of the burn points. First, a case is defined by requiring all the burn points placed on the nominal path. This is called the 'Special Case'. Next, a case is defined by allowing the burn points to be placed anywhere within the constraint volume, called the 'General Case'. The special case may be required if the spacecraft is required to be a constant distance from the chief. For instance, there may be a plume exclusion zone or distance requirement.

The general case has the most operational utility, because remote measurements of the chief will be required after the relatively dynamic behavior of the spacecraft settles

after each subsequent burn point. Because the burn points do not have to be on the nominal path, it allows more flexibility in choosing where burns are placed.

Overview of Content

A method involving discrete impulsive burns is desired. These maneuvers require significant propellant expenditure to achieve the required circumnavigation trajectories within the desired TOF. The placement, relative to the chief satellite, and the timing of these discrete impulses (at the burn points) has a significant impact on the propellant required for a given maneuver.

Hill's equations are used as the primary tool for modeling the dynamics for the required maneuvers. The equations are used to calculate the total impulse, Δv_t, required for a given circumnavigation maneuver. From these equations, the position, magnitude, and direction of each discrete impulse, Δv, is determined. The magnitudes are summed to determine the total impulse required. The required impulse is directly proportional to the amount of thrust a propulsion system must provide, and thus the mass of propellant required for a given mission.

A simple method for placement of the burn points is initially developed, and used as an initial guess for numerical optimization. The optimization routine is used to investigate the lowest minimum propellant required. From analyzing the optimization results, an analytical algorithm is proposed to approximate the minimum total required impulse for a circumnavigation maneuver, and to develop a more robust initial guess for the numerical optimization. This algorithm's performance is then evaluated for several cases.

II Methodology

Assumptions

The chief or RSO is assumed to be in a circular orbit, with the deputy orbit having a very small eccentricity. Additionally, a two body, point-mass gravitational model is assumed (i.e. no perturbations), and the distance between the deputy and the chief is much less than the radius of the chief's orbit. These assumptions allow the use of Hill's equations for relative orbital motion.

Hill's Equations.

A specific form of Hill's equations (16:83) is used (also known as the complete Clohessy-Wiltshire solution) and shown as:

$$\begin{bmatrix} d\vec{r}(t) \\ d\vec{v}(t) \end{bmatrix} = \Phi(t-t_0)\begin{bmatrix} d\vec{r}(t_0) \\ d\vec{v}(t_0) \end{bmatrix} \tag{3}$$

The matrix (Φ) (16:83) are defined as:

$$\Phi(\Delta t) = \begin{bmatrix} \Phi_{rr} & \Phi_{rv} \\ \Phi_{vr} & \Phi_{vv} \end{bmatrix} = \begin{bmatrix} 4-3\cos(n\Delta t) & 0 & 0 & \frac{1}{n}\sin(n\Delta t) & \frac{2}{n}(1-\cos(n\Delta t)) & 0 \\ 6(\sin(n\Delta t)-n\Delta t) & 1 & 0 & \frac{2}{n}(\cos(n\Delta t)-1) & \frac{4}{n}\sin(n\Delta t)-3n\Delta t & 0 \\ 0 & 0 & \cos(n\Delta t) & 0 & 0 & \frac{1}{n}\sin(n\Delta t) \\ 3n\sin(n\Delta t) & 0 & 0 & \cos(n\Delta t) & 2\sin(n\Delta t) & 0 \\ 6n(\cos(n\Delta t)-1) & 0 & 0 & -2\sin(n\Delta t) & -3+4\cos(n\Delta t) & 0 \\ 0 & 0 & -n\sin(n\Delta t) & 0 & 0 & \cos(n\Delta t) \end{bmatrix} \tag{4}$$

where n is the mean motion of the chief's orbit and $\Delta t = t - t_0$. Equation 3 determines the position, $\delta\mathbf{r}$, and velocity, $\delta\mathbf{v}$, relative to the chief at a time, *Dt*, later than the initial position and velocity.

Assumptions in Initial Conditions.

The deputy's initial position and velocity relative to the chief are assumed to be known. However, in order to generalize the results, zero initial relative velocity is used for all calculations. Zero initial velocity ensures no component direction of the initial velocity can subtract or add to the impulse at the initial point. This essentially cancels the effects of any variations in the initial conditions on the overall optimization.

Additionally, the deputy is required to not exceed a given distance from the nominal path during any part of the maneuver. The distance from the deputy to the nominal path is defined as a maximum deviation, ρ_{max}. This deviation, ρ_{max}, is measured as the magnitude of the spatial deviation vector of the flight path from the nominal path. This spatial deviation vector is thought of as a radius from the nominal path, thus ρ_{max} is termed the maximum deviation radius. The actual deviation (of the flight path from the nominal path) is only measured in a spatial sense. It does not take into account when and where the deputy is located on the flight path with reference to when and where the deputy is to be nominally located along the nominal path. The ρ_{max} constraint defines a toroidal constraint surface about the nominal path.

Instantaneous Dv Assumption.

Instantaneous impulses, Δv, which occur at discrete points in space, are assumed. This assumption is less valid for low thrust vehicles or for extremely fast circumnavigation times of flight when impulses may require a significant amount of time to impart a change in velocity. For instance, this assumption breaks down as the

individual maneuver durations become a significant fraction of the circumnavigation time of flight, TOF.

Nominal Path

The orientation and size of the circular nominal path can be described by four parameters: r_0, ?, T_y, and T_z. (4:3) A 2-3 space fixed T_y, T_z rotation sequence of a circle of radius, r_0, in the y-z plane defines the nominal circular path. The angle ? defines a spatial degree of freedom along the circular path with the initial point being defined by γ_0. Figure 1 illustrates this rotation. The values of T_y, T_z, and r_o are assumed to be given quantities, whereas ? is a variable which must be varied to determine points on the circle.

These four parameters are defined in the Local Vertical, Local Horizontal (LVLH) coordinate system. The LVLH coordinates define the y direction in the same direction of the chief's instantaneous velocity vector. The x direction is defined in the radial (from the center of the Earth) direction to the chief, and consequently the z direction is orthogonal to x and y. This coordinate system is equivalent to the RSW coordinates used in many texts (14:162-163).

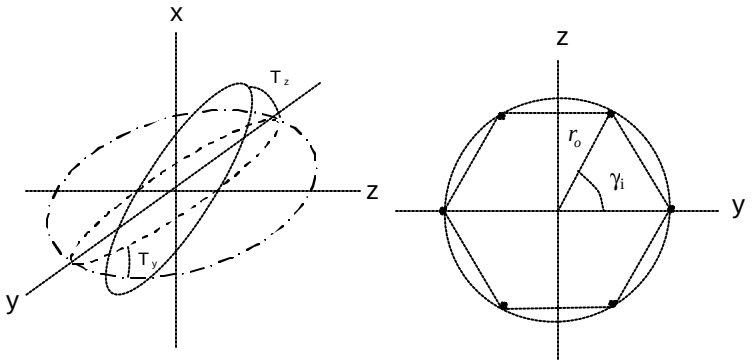


Figure 1 a) Rotated Nominal Path b) Unrotated Nominal Path

Mathematically, any point along the nominal path is represented by a phasing angle, γ_i, and by rotating an initial vector of length r_0 placed along the y direction ([0; r_0; 0]). This corresponds to the initial point located with $\gamma_0 = 0$. Using the rotation described above, the position vector as a function of γ is (4:3)

$$d\vec{r}(g_i) = \begin{bmatrix} r_0 \cdot \cos(\Theta_z) \cdot \sin(\Theta_y) \cdot \sin(g_i) - r_0 \cdot \sin(\Theta_z) \cdot \cos(g_i) \\ r_0 \cdot \cos(\Theta_z) \cdot \cos(g_i) + r_0 \cdot \sin(g_i) \cdot \sin(\Theta_z) \cdot \sin(\Theta_y) \\ r_0 \cdot \cos(\Theta_y) \cdot \sin(g_i) \end{bmatrix} \quad (5)$$

where $\gamma_0 = 0$.

State Vector Definition

Unique spatial positions where discrete, instantaneous impulses occur are called 'burn points'. These burn points are required to perform the circumnavigation within the required total time. Assigning individual time of flights between them allows for the computation of the Δv required at each burn point. Hill's equations (16:80) were used to determine the total impulse, Δv_t, required for a particular maneuver. This parameterization assumes the time, t_i, at each i^{th} point is known; the times, t_i, are independent variables.

A complete circumnavigation is defined by a 2π rotation in γ from some given initial position (defined by a γ_o on the nominal path) within the required total time of flight (TOF). The parameter, *b*, indicates the total number of discrete burn points along the circumnavigation path.

A state vector, X, is composed of the spatial degrees of freedom for the burn points' positions, and the corresponding times when the deputy is located at the burn point positions. There are two probable cases investigated for constraining the placement of the burn points. First, the *'special case'* requires all the burn points to be placed on the

nominal path. Therefore, the special case only requires one degree of freedom, γ_i, to define a burn point placement. Next, the *'general case'* is defined by allowing the burn points to be placed anywhere within the constraint volume. For both cases, the burn point timing is not specified, only the sum of the times as defined by the circumnavigation.

Special Case State Vector.

The special case state vector is built from the discrete values of γ_i and t_i, defined by

$$X = \begin{bmatrix} g_1 \\ \vdots \\ g_i \\ t_1 \\ \vdots \\ t_i \end{bmatrix}, \; for \; i = 1,...,b-1 \tag{6}$$

where b is the total number of burn points. In order to assure the circumnavigation is complete (i.e. the deputy returns to the initial point), the angle to the final position, γ_b, is computed as 2π minus the sum of the previous γ_i's. Similarly, the time of the final point, t_b, is computed as TOF minus the sum of all the previous t's. These values are computed as

$$\begin{aligned} g_b &= 2p - \sum_{j=1}^{b-1} X(j) \\ t_b &= TOF - \sum_{j=1}^{b-1} X(j+b-1) \end{aligned} \tag{7}$$

General Case State Vector.

A general case, representing three degrees of spatial freedom, defines the burn points placed within a solid torus whose minor radius is defined by ρ_{max}. Any point along the actual flight path (and within the solid torus) has a vector from it to the nominal path which represents a deviation radius, and has a magnitude of ρ. This radius is rotated about the nominal path by an angle, e as shown in Figure 2. The angle e can be rotated through 2p defining a circle about any point on the nominal path. Rotating this circle by ?, creates a torus with an inner radius, $r_c = r_0-\rho$, and an outer radius, $r_t = r_0 + \rho$. The inner and outer radii will be used in Chapter V below.

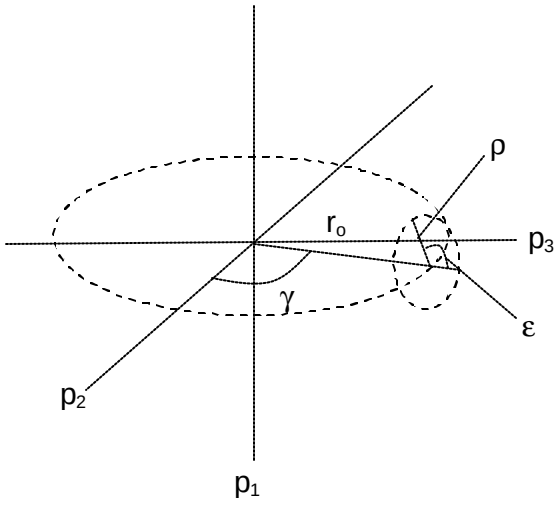


Figure 2. Sketch of Torus Parameterization

Figure 2 gives an illustration of the torus parameters to define a unique point within the torus. A coordinate frame is fixed in the torus where the p_2 direction is defined by the position of the initial position, which can be defined by the initial angle, γ_0, with respect to the LVLH frame. The p_3 direction lies within the nominal path plane

orthogonal to p_2, and p_1 completes the triad. The radius, ρ, is allowed to vary from 0 to ρ_{max}.

Any point in the solid torus can be given by

$$x_p\hat{p}_1 + y_p\hat{p}_2 + z_p\hat{p}_3 = r\sin(e)\,\hat{p}_1 + (r_0 + r\cos(e)\cos(g))\hat{p}_2 + (r_0 + r\cos(e)\sin(g))\,\hat{p}_3 \quad (8)$$

where x_p, y_p, and z_p are components in the path coordinate system [p_1 p_2 p_3]. Equation (8) is modified from a general torus parameterization. (8) The position vector is expressed in LVLH coordinates using the nominal path rotation angles Θ_y and Θ_z discussed above. Thus the position vector, $\delta\mathbf{r}$, is defined in the LVLH frame as

$$d\vec{r}(r,e,g) = \begin{bmatrix} (r\sin(e))\cos(\Theta_y)\cos(\Theta_z) + (r_o + r\cos(e))\sin(g)\cos(\Theta_z)\sin(\Theta_y) - (r_o + r\cos(e))\cos(g)\sin(\Theta_z) \\ (r_o + r\cos(e))\cos(g)\cos(\Theta_z) + (r\sin(e))\cos(\Theta_y)\sin(\Theta_z) + (r_o + r\cos(e))\sin(g)\sin(\Theta_y)\sin(\Theta_z) \\ (r_o + r\cos(e))\sin(g)\cos(\Theta_y) - (r\sin(e))\sin(\Theta_y) \end{bmatrix} \quad (9)$$

where γ_0 is set to zero.

Now the general case state vector is defined using the three degrees of freedom:

$$X = \begin{bmatrix} r_{d1} \\ \vdots \\ r_{di+1} \\ e_1 \\ \vdots \\ e_{i+1} \\ g_1 \\ \vdots \\ g_i \\ t_1 \\ \vdots \\ t_i \end{bmatrix}, \; for \; i = 1,\ldots, b-1 \quad (10)$$

where the final point in-plane angle, γ_b, and time, t_b, are computed as in Eq. (7), but the final point radius, ρ_f, and torus angle, e_f, are allowed to vary.

Total Impulse Computation (Cost Function).

To compute each discrete individual Δv requires knowledge of the individual time of flight between the i^{th} and the $(i+1)^{th}$ burn point, calculated as

$$\Delta t_1 = t_{i+1} - t_i \qquad \Delta t_2 = t_i - t_{i-1} \tag{11}$$

Once the position of the current burn point, $\delta\mathbf{r}(t_i)$, along with the time of flight, Δt_1, to the next burn point, $\delta\mathbf{r}(t_{i+1})$, is known; the required velocity, $\delta\mathbf{v}^+(t_i)$, immediately following the burn is computed as

$$d\vec{v}^{+}(t_i) = \Phi_{rv}^{-1} \cdot \left[\Phi_{rr}(\Delta t_1) \cdot d\vec{r}(t_i) - d\vec{r}(t_{i+1})\right] \tag{12}$$

The velocity just prior to the burn, $\delta\mathbf{v}^-(t_i)$, is calculated by

$$d\vec{v}^{-}(t_i) = \Phi_{vr}(\Delta t_2) \cdot d\vec{r}(t_{i-1}) + \Phi_{vv}(\Delta t_2) \cdot d\vec{v}^{+}(t_{i-1}) \tag{13}$$

where Φ_{vr} and Φ_{vv} are defined above in Eq. (4).(16:80) This velocity is determined by the location and magnitude of the previous Δv. Eq. (13) is valid for all burn points except for the initial one. At the initial burn point, $\delta\mathbf{v}^-(t_i)$ is assumed to be zero for all calculations.

Once the velocity just prior to the burn point and the velocity just after the impulsive burn is known, the Δv magnitude and direction is computed as

$$\Delta\vec{v}_i = d\vec{v}^{+}(t_i) - d\vec{v}^{-}(t_i) \tag{14}$$

Now that the individual Δv vectors are computed, the total required impulse can be minimized. The total impulse is given by

$$F(X) = \Delta v_t = \sum_{i=0}^{b-1} \left\| \Delta \vec{v}_i \right\| \tag{15}$$

Singularity in Cost Function.

The cost function (Eq. (15)) is not continuous within the feasible region of most cases. This is a result of using the inverse of the Φ_{rv} matrix in Eq. (12); which is computed as

$$\Phi rv^{-1}(\Delta t_i) = \begin{bmatrix} \frac{n(3y - 4\sin(y))}{-8 + 8\cos(y) + 3y\ \sin(y)} & \frac{-2n(\cos(y) - 1)}{-8 + 8\cos(y) + 3y\ \sin(y)} & 0 \\ \frac{2n(\cos(y) - 1)}{-8 + 8\cos(y) + 3y\ \sin(y)} & \frac{n\sin(y) - 1)}{8 - 8\cos(y) - 3y\ \sin(y)} & 0 \\ 0 & 0 & \frac{n}{\sin(y)} \end{bmatrix} \tag{16}$$

where $\psi = n\ t$. This matrix contains a singularity when ψ is zero or an integer multiple of π. The cost function's gradient becomes very steep in the region near the singularity, and the numerical optimization routine will not converge to a solution if the search routine approaches the singularity. Physically, this singularity is represented by burning between two points of a finite distance apart in zero time which requires infinite Δv.

Several strategies are employed to mitigate the singularity's effects on the numerical optimization. These strategies include initial guess inputs into the optimization close to a local minimum, utilizing a low number of burn points in the initial guess, searching over different potential numbers of burn points, and establishing bounds on the time instances to be $\varepsilon < t_i$. In practice, this lower limit of the times, ε, has been set to $(1x10^{-7})$ x TOF to prevent the routine from approaching too close to the singularity.

These mitigations allow good performance of the optimization routine in the vicinity of a local minimum.

Flight Path Constraint

As mentioned previously, the actual circumnavigation flight path between burn points must be contained within the torus and, therefore, must not deviate from the nominal path by more than ρ_{max}. For this reason, the deviation from the nominal path must be calculated to ensure the relative satellite trajectory meets this constraint. The flight path between burn points is calculated in discrete time steps by propagating Hill's equations (16:80) forward after the burn is applied:

$$d\vec{r}(t_{\text{int}}) = \Phi_{rr}(\Delta t_{\text{int}}) \cdot d\vec{r}(t_i) + \Phi_{rv}(\Delta t_{\text{int}}) \cdot d\vec{v}^{+}(t_i) \tag{17}$$

where Δt_{int} are intermediate time steps defined by dividing Δt_1 by z time steps. A value of z = 20 was chosen for all calculations. This value of z provided adequate resolution of the path's shape and magnitude. The intermediate position vector, $\delta\mathbf{r}(t_{int})$, is used to determine the flight path deviation represented as vector, r_{dev}, illustrated in Figure 3.

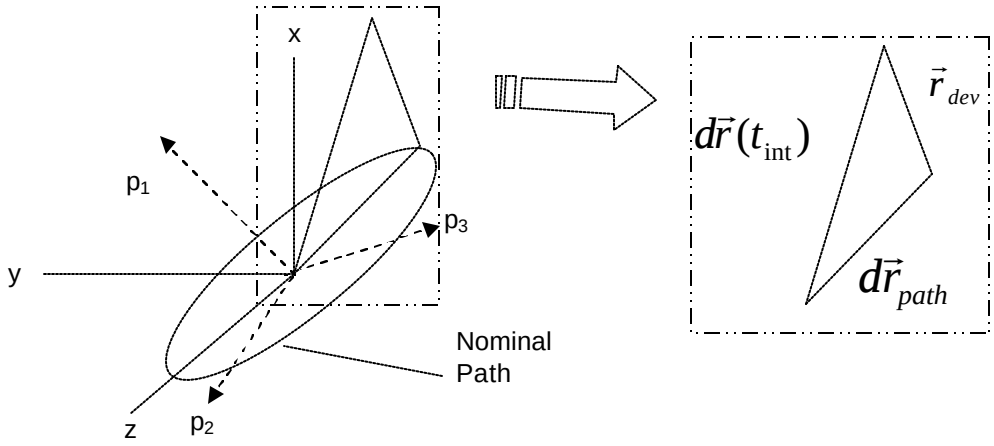


Figure 3. Path Constraint Definition Sketch

The 'path' frame [p1 p2 p3] is defined by the space-fixed 1-2-3 rotation by γ_o, Θ_y, and Θ_z from the LVLH frame. Assuming the initial position always lies on the nominal path, the corresponding transformation matrix, *R*, from the path frame to the LVLH coordinates is

$$R = \begin{bmatrix} \cos(\Theta_y)\cos(\Theta_z) & \cos(\Theta_y)\sin(\Theta_z) & -\sin(\Theta_y) \\ \cos(\Theta_z)\sin(\Theta_y)\sin(g_0) - \cos(g_0)\sin(\Theta_z) & \cos(g_0)\cos(\Theta_z) + \sin(g_0)\sin(\Theta_y)\sin(\Theta_z) & \cos(\Theta_y)\sin(g_0) \\ \cos(\Theta_z)\sin(\Theta_y)\cos(g_0) + \sin(g_0)\sin(\Theta_z) & -\sin(g_0)\cos(\Theta_z) - \cos(g_0)\sin(\Theta_y)\sin(\Theta_z) & \cos(g_0)\cos(\Theta_y) \end{bmatrix} \quad (18)$$

This matrix, *R*, is then used in the subsequent Eqs. (19), (20), and (21). Each intermediate point is transformed into the path frame by

$$\begin{bmatrix} dp_1 \\ dp_2 \\ dp_3 \end{bmatrix} = R \cdot d\vec{r}(t_{\text{int}}) \quad (19)$$

The intermediate position vector, $d\mathbf{r}(t_{int})$, is projected into the plane of the nominal path to calculate the position vector of the nominal path's closest point. Since

there is no preferred timing of the intermediate points along the circumnavigation, the closes point is found as

$$d\vec{r}_{path} = \frac{r_0 \begin{bmatrix} dp_1 \\ dp_2 \\ 0 \end{bmatrix}}{\sqrt{dp_1{}^2 + dp_2{}^2}} \tag{20}$$

The flight path deviation vector, r_{dev}, is calculated by differencing the intermediate position vector and the projected path vector in the path frame:

$$\vec{r}_{dev} = (R \cdot d\vec{r}(t_{\text{int}})) - d\vec{r}_{path} \tag{21}$$

Finally, the flight path constraint is defined by deviation radius magnitude, $\rho_{dev,}$ which cannot exceed ρ_{max}. The constraint is computed by ensuring the difference is never positive:

$$\left|\vec{r}_{dev}\right| - r_{\max} \leq 0 \tag{22}$$

Additional Constraints

Each burn point's γ and t cannot exceed 2p and TOF respectively from the problem statement above. Additionally, the values for γ and t must be zero or greater (positive). These values are used to define the upper, X_u, and lower, X_l, limits on the state vector.

A linear inequality constraint is needed to ensure γ_b and t_b are not negative. From Eq. (7), it can be seen the sum of the γ's and the sum of the t's cannot be greater than 2π or TOF respectively. Physically, this would represent the circumnavigation doubling back on itself producing a negative γ_b, and time flowing backwards giving a negative t_b.

Numerical Optimization

A numerical optimization technique is employed to locate locally minimum fuel trajectories. The general form of the optimization problem can be represented as

$$\begin{aligned} &\min_{X \in R^n} F(X) \\ &G_{cineq}(X) \leq 0 \\ &X_l \leq X \leq X_u \end{aligned} \tag{23}$$

where $G_{cineq}(X)$ represents the nonlinear constraints from Eq. (22) and linear inequality constraints from above. The states, X_l and X_u, represent the lower and upper bound constraints on the state vector. The goal is to find the state vector producing the minimum value of the cost function, F(X) computed in Eq. (15).

The cost function, F(X), is highly non-linear. The '*fmincon*' function in MATLAB's Optimization Toolbox (6) was chosen to perform the optimization. This routine is designed using Sequential Quadratic Programming (9:3-26) which allows for the use of nonlinear constraints, is appropriate for a single objective cost function, and allows for linear constraints as well. It is limited by the fact the cost function must be continuous over the interval, and will also attempt to minimize the maximum constraint if there is no feasible solution (9:3-26).

MATLAB Optimization Routine.

As mentioned above, MATLAB's '*fmincon*' function uses the Sequential Quadratic Programming method which is composed of three main steps: updating the Hessian matrix of the Lagrangian equation, solving the Quadratic Programming sub-problem, and performing a Line-search and Merit function calculation. (9:Sec.3, 26).

The optimization problem can be reformulated into the Kuhn-Tucker Equations (9: Sec.3, 26):

$$\nabla F(X^*) + \sum_{i=1}^{m} (l_i^* \cdot \nabla G_{cineq}(X^*)) = 0$$
$$l_i^* \cdot G_{cineq}(X^*) \geq 0 \tag{24}$$

where λ_i^* is a Lagrange multiplier termed a Kuhn-Tucker point at a unique state vector, X^*, and *m* is the number of constraints. This equation essentially balances the gradients of the active constraints and the gradient of the cost function to find a minimum. If the minimum lies on the constraint boundary, it may not be a true minimum, but the least cost function value along the boundary. The Kuhn-Tucker points can be found by solving the Lagrangian equation (9: Sec.3, 27):

$$L(X, l) = F(X) + \sum_{i=1}^{m} (l_i \cdot G_{cineq}(X)) \tag{25}$$

The algorithm starts with an initial guess state vector. From the initial guess, a Hessian is computed using finite difference calculations. The Quadratic Sub-Problem is then solved (9: Sec.3, 28 and 2:238) to determine the search direction. Once the search direction has been defined a line search and merit function are used to determine the step size in order to update the state vector. The updated state vector is:

$$X_{k+1} = X_k + a d_k \tag{26}$$

where d_k gives the search direction and α is the distance along the search direction. Once the step size is determined the gradient of the function is evaluated at the new point, and evaluated against Eq. (24). If the convergence criteria are not met, the BFGS (Broyden-

Fletcher-Goldfarb-Shanno) method (1:330 and 9: Sec.3, 30) is then used to determine an updated Hessian matrix, and the procedure is reiterated.

Practical MATLAB Usage.

MATLAB uses finite differencing techniques to numerically compute the gradient and Hessian of the cost function at any given state vector. These finite differences require bounds on the step size for the finite difference. The default was set at $1x10^{-8}$, but $1x10^{-9}$ produced higher quality results. Additionally, the tolerance on the state vector, the cost function and the constraints can be set as well. Changing these tolerances produced significant differences in the output of the optimization program. Setting all tolerances equal to $1x10^{-9}$ produced the most consistent results for all the cases presented.

Functionally, the state vector quantities were normalized for input into MATLAB's '*fmincon*' routine. This gives a similar magnitude between the states and produced better convergence. The angles ε and γ were normalized by 2π to give values between zero and one , the deviation, ρ was normalized by the maximum deviation radius, ρ_{max}, and time was normalized by TOF. (Angles of the special case shown below were not normalized.)

Optimization Results Check.

In order to check the optimization program, the cost function is evaluated in select directions around the area of the minimum given by the program. The goal of the check is to gain confidence in the optimization results and understand what levels in the numerical tolerances produced the best results. This method is based upon a subset of the Weierstrass Theorem. (1:83)

Specifically, the cost function is evaluated at the optimized state vector stepped in only one state by a range of finite steps and does not utilize the full set of possible states as required by the Weierstrass Theorem. Each step creates a new state vector, X':

$$X' = X + g_j(m) \tag{27}$$

where m is the step size, and g is given as

$$g_j(m) = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ m \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{28}$$

where j represents the state to be stepped. This step vector is the same length as the original state vector, X.

If the optimized state vector is to produce a local minimum within the defined tolerances, the new stepped value of the cost function, F(X'), cannot be less than the cost function value from the optimization output:

$$F(X') - F(X) > 0 \tag{29}$$

This step method described doesn't fully ensure a local minimum has been found because the cost function could decrease in a direction not orthogonal to the states. For instance, simultaneously stepping states 1 and 2 by some amount produces another unique state vector and thus another unique cost function value. This check describes a necessary condition for a minimum, but doesn't describe a sufficient condition.

If given a stepped value of the cost function for which Eq. (29) is not satisfied, the state X does not represent a minimum in the cost function. However, the new state vector must be evaluated to determine if it meets the constraints as well. If the new state does not meet the constraints, it is not a valid state, and therefore must be disregarded.

Functionally the new stepped state is compared to the constraints. If the new path violates constraints an integer number is added to a variable called the *check sum*. This *check sum* is zero if no constraints were violated, and greater than zero if the constraints were violated. The other constraint's violations (upper and lower limits of the state vector and the linear inequality constraint on the sum of the times and γ's) were included in this *check sum* as well.

An alternate way to verify a minimum is to use the Kuhn-Tucker necessary conditions (1:122). The Kuhn-Tucker method requires the calculation of the gradient of the cost function at the specific points. However, this method was not needed since the step method described above was able to provide enough confidence in the minima found by the optimization routine.

III. Equal Angle/Equal Time Method

The behavior of the cost function is initially evaluated with a very simple analytic method in defining the placement and timing of the individual Δv_i's. This method provides a baseline for comparing the optimization's or other analytic methods' performance. The Equal Angle/Equal Time (EAET) method is defined by the burn points placed upon the nominal path in equal angular displacements along the nominal path and spaced equal times apart. Each angular location and time is given by

$$\gamma_i = \frac{2p}{b}, \quad t_i = \frac{TOF}{b} \tag{30}$$

where *b*, the number of burn points, must be specified.

Figure 4 shows the behavior of the EAET with the TOF set to 0.1 times the orbital period of the chief (about 9.25 minutes). The chief is in a circular orbit with an altitude of 400 km for all calculations and results shown throughout this thesis, unless otherwise stated. The nominal path is oriented in the x-y plane ($\Theta_y = 90°$ and $\Theta_z = 0°$) with the initial point rotated 45° along the path ($\gamma_0 = 45°$). The nominal path's radius is set at 50 m, with the deviation constraint, ρ_{max}, equal to 10 m. Two paths are shown in Figure 4. The 'Min Actual Path' is using five burn points (b = 5), and the 'Infeasible Path' is using four burn points (b = 4). The constraint surface is shown as the circular grid, representing the torus's surface.

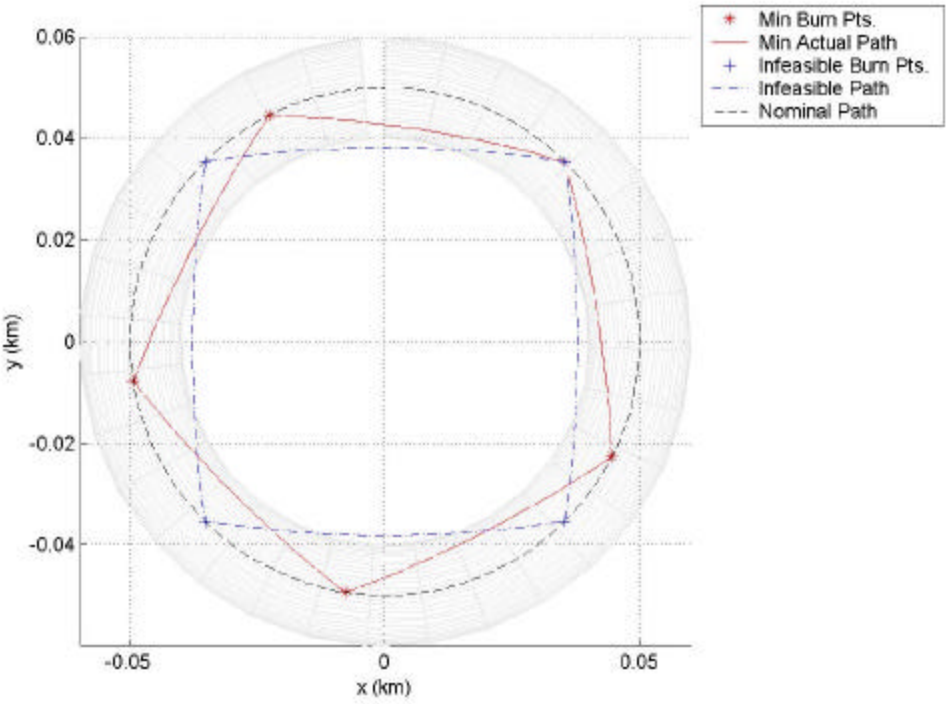


Figure 4. EAET with Minimum Feasible Number of Burn Points (b = 5). $\Theta_y = 90^o, \Theta_z = 0^o, \gamma_o = 45^o$, TOF = 0.1, and ρ_{max} = 10 m

Minimum Number of Burn Points for Given r_{max}.

For the EAET method there exists a minimum b for the required ρ_{max} constraint to be satisfied. The deviation from the nominal path is apparent, and depends upon the b used in Equation 30. The number of burn points, b, is stepped in integer increments from 2 until the minimum number of burn points case meets the flight path constraint. In Figure 4, the b = 4 case exceeds the maximum deviation.

Dv_t Versus Number of Burn Points.

The difference in magnitude of Δv_t (also called the change in velocity in some Figures) between the continuous and the discrete methods of circumnavigation can be quantified by Figure 5. Figure 5 shows the variation of Δv_t as a function of the number of burn points for several path orientations. The circumnavigation for each case shown has its nominal path rotation $\Theta_z = 0^o$, a radius of 50 meters and a TOF = 0.1 times the chief's period (555 s).

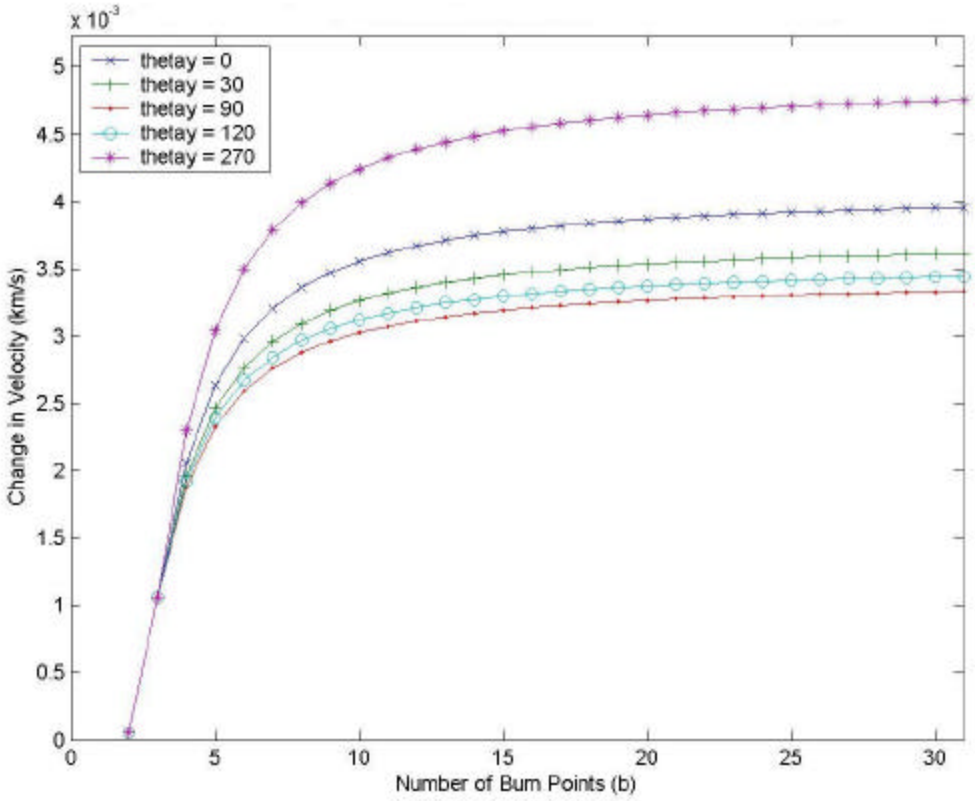


Figure 5. Δv_t Versus the Number of Burn Points.

The flight path's deviation from the nominal path increases as the number of burn points decreases, which can be seen qualitatively from Figure 4. As the number of burn

points increases the value of Δv_t approaches a value that it would be for a continuous burn on the nominal path.

The flight path's deviation from the nominal path is also a function of the number of burn points. Therefore, the value of ρ_{max} determines the minimum number of burn points that produce a trajectory within the torus. The minimum number of burn points is determined by trial-and-error as above and represents the minimum Δv_t for the EAET method.

The EAET method is a very simple algorithm allowing for a quick determination of the order of the required Δv_t, but further investigations show room to further minimize Δv_t. However, due to its simplicity, the EAET method presents a good basis for measuring the performance of the optimization results. The Δv_t for each subsequent optimization result and analytical design method is compared to the EAET by computing the percentage of savings from the EAET method for the given circumnavigation.

Comparison with Continuous Control Method.

The EAET method allows a direct comparison to continuous control techniques. (4) The paradigm used differs from continuous control technique by the allowance of the intermediate flight path between burn points to vary off of the defined nominal path. The continuous control paradigm ensures all points on the circumnavigation follow the nominal path vice the discrete methods proposed require the path to lie within a constrained region about the nominal circular path. The variance or deviation from the nominal path allows for considerable savings in the Δv_t.

Varying the nominal path's orientation and TOF also affects Δv_t required. The Δv_t varies similarly with TOF, Θ_y, and Θ_z as found in the continuous method (4:3) except the magnitude of the Δv_t is less. For instance, Figures 6 and 7 demonstrate the variation of Δv_t with rotating a 50 m nominal path about Θ_y while varying TOF from 0.1 to 1. The graphs show the Δv_t variation as computed by the EAET method with six burn points (b=6).

The shapes of the curves are the same as demonstrated in the continuous control method (4:6) with a few exceptions. The overall magnitude of the surface is less than the magnitude presented in the continuous case, except at the minimum points. Also, because the initial velocity is set to zero relative to the chief, a finite amount of Δv is required to put the circumnavigation on a natural motion trajectory which occurs at Θ_y = 30º and 120º with a TOF = 1 and Θ_z = 0º. (11:7) For these cases, the continuous case Δv_t would be zero as can be seen from Eq. (2) and as presented in Reference 4. (4:6) However, the minimums and maximums still occur at the same respective values of Θ_y, Θ_z, and TOF.

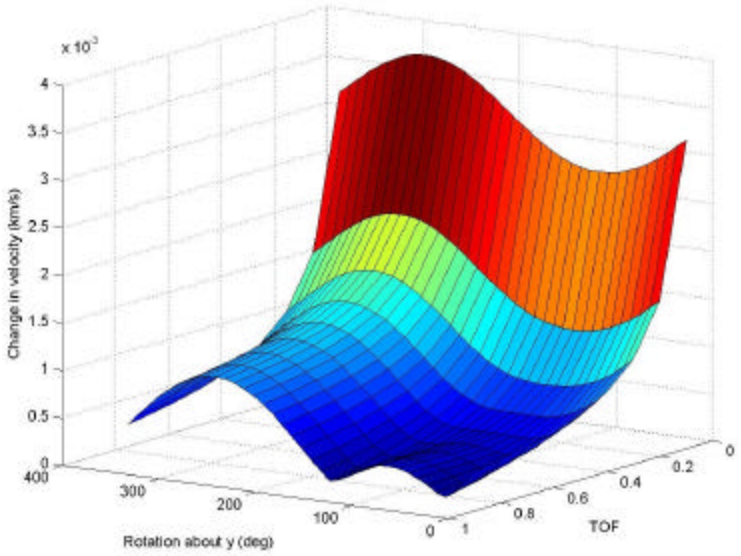


Figure 6. Δv_t Surface From Varying TOF and Θ_y

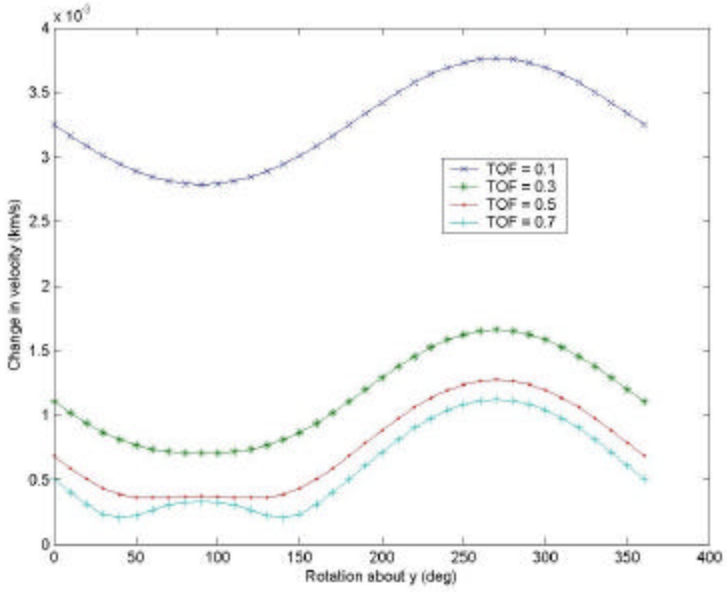


Figure 7. Δv_t Cross-Sections From Varying TOF and Θ_y

This comparison is extrapolated into the results for the optimization cases, and subsequent analytical methods. For cases with TOF less than 0.5 times the chief's orbital period, the minimum Δv_t occurs at $\Theta_y = 90^o$ and $\Theta_z = 0^o$. This case will be the primary example for the lower total impulse trajectories calculated below.

IV. Optimization Results

The EAET method has two drawbacks: the Δv_t is not optimal and the intermediate flight path constraints cannot be enforced except by trial-and-error selection of the number of burn points. To investigate the behavior of optimal maneuvers that satisfy the path constraint, we use numerical optimization. The optimization results are presented using the two defined cases: the *special case* and the *general case*.

Special Case Results

The EAET method is used as the initial guess in the optimization of the cost function shown in Eq. (15). Several other types of guesses were investigated, but they mainly involved random choices of the states, and did not provide lower cost function values than the EAET method. Figure 8 shows the results from using the EAET initial guess with b = 5, ρ_{max} = 10 m and a TOF = 0.1. The nominal path is oriented in the x-y plane to allow for simpler viewing of the actual flight paths; additionally, the x-y plane represents the minimum required circumnavigation for this TOF as shown in Figure 7. The initial guess's (EAET's) value for the Δv_t is 2.6082 m/s for the path defined in Figure 8. The final optimized Δv_t value is 2.4485 m/s and represents a 4.55% savings in Δv_t.

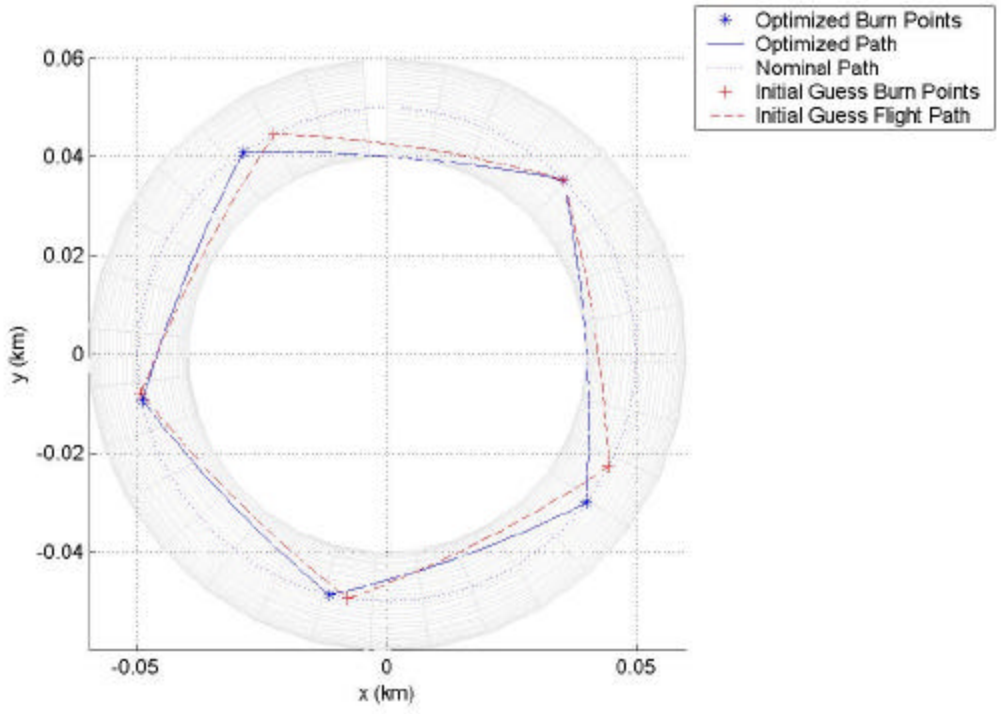


Figure 8. Five Burn Points, $\Theta_y = 90^o$, $\Theta_z = 0^o$, $\gamma_o = 45^o$ with TOF = 0.1 and $\rho_{max} = 0.01$ km

One fundamental difference between the guess and the optimized solution is the deviation of the intermediate flight path from the nominal. Figure 9 shows the magnitude the deviation from the nominal path, ρ_{dev}, for the both the initial guess and the optimized solution. The zero points for both lines in Figure 9 represent the burn points, which are placed upon the nominal path. Note the EAET flight path does not directly touch the constraint surface.

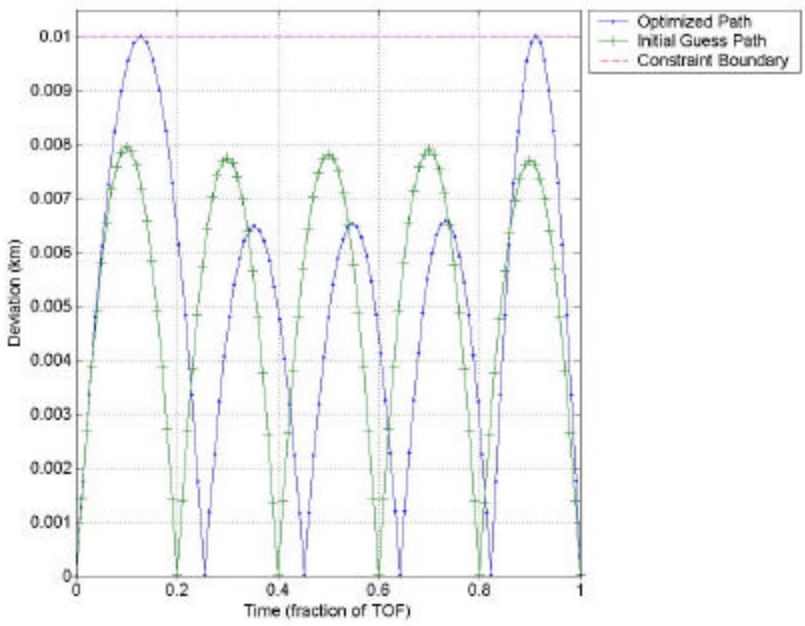


Figure 9. Five Burn Points Deviation, $\Theta_y = 90^o$, $\Theta_z = 0^o$, $\gamma_o = 45^o$ with TOF = 0.1, and $\rho_{max} = 0.01$ km

The optimized intermediate flight path touches the constraint boundary after the initial burn and the final burn as seen in Figure 9; this excursion to the constraint boundary skirts the inner radius of the torus as seen in Figure 8. The optimization consistently found minima where the intermediate flight path skirts the constraint boundary. This skirting is characteristic of all the reasonable optimal solutions computed.

The next step is to investigate the effect of varying the flight path constraint, $\rho_{max.}$ Figure 10 shows the optimization for a path with the same orientation and TOF as Figure 8, but with $\rho_{max} = 20$ m. The minimum number of burn points for the EAET case is now four. The optimized Δv_t was determined as 2.05 m/s which represents a 12.5% savings over the EAET value of 2.34 m/s.

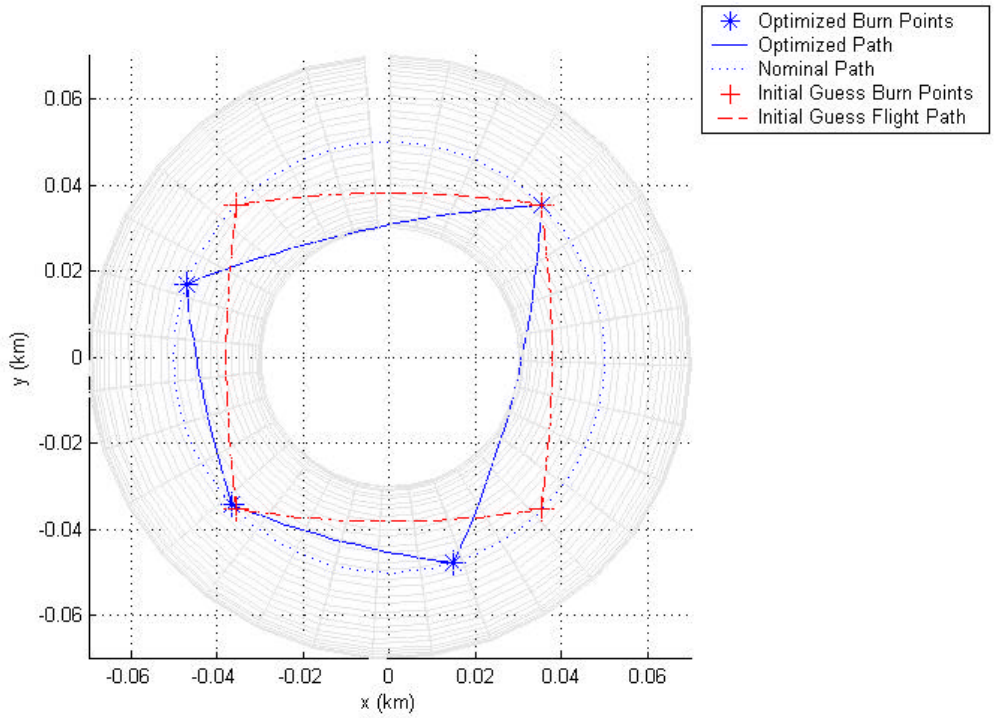


Figure 10. Four Burn Points, Decrease Constraint: $\Theta_y = 90^o$, $\Theta_z = 0^o$, $\gamma_o = 45^o$, TOF = 0.1 and $\rho_{max} = 0.02$ km

The optimization routine placed the first and last burn points such that the first and last paths were tangential to the inner constraint surface. This was a common theme while decreasing the constraint (increasing ρ_{max}). Ultimately, increasing ρ_{max} decreases the total number of points possible, and if the maximum deviation is large enough the circumnavigation requires only two burns which represents the minimum number of burn points for a reasonable circumnavigation.

Optimization Results Check Output.

The optimization results check as described above is used to determine whether or not the optimization routine found a local minimum. A variety of steps sizes were utilized ranging from 1×10^{-10} to 1×10^{-1} step sizes in increments of one order of magnitude. Each stepped state vector is evaluated to ensure the circumnavigation constraints are met.

To show whether the constraints were met or exceeded for each new stepped state vector, a Violation Matrix is developed. The Matrix is calculated for each new stepped state vector, X', and is a row vector with six columns. If the value in the column is zero then the particular constraint corresponding to the column was met, likewise if the value is greater than zero that constraint has been violated the integer value number of times. The constraint definitions are given in Table 1.

Table 1. Violation Matrix Definition

Violation Matrix Column Number	**Constraint**
1	$\sum_{i}^{b-1} g_i < 2p$
2	$\sum_{i}^{b-1} t_i < 1,$
3	$\gamma_i < 2\pi$
4	$t_i < 1$
5	γ_i and $t_i > 0$
6	$\left\|\vec{r}_{dev}\right\| - r_{\max} \leq 0$

Table 2 shows part of the step check data from the circumnavigation and optimization presented in Figure 8 above. The full data are presented in Appendix A. The top half of Table 2 for a step size of $1x10^{-7}$ produces changes in the cost function much lower than the cost function tolerance; in this case the tolerance on the cost function was set at $1x10^{-9}$. The lower half (bolded numbers) represent the step size which shows the optimization meets the local minimum requirements. If the Step Check column is negative, then the step size and direction represent a more minimum value for the stepped cost function. However, since the optimization result was constrained the bolded negative values all exceed constraints and therefore are not feasible states. The feasible states are all positive. This is true for the step size of $1x10^{-6}$, but decreasing the step size to $1x10^{-7}$ represents a case where the step size produces a difference in the new cost function two orders below the numerical limit of the optimization criteria.

Table 2. Special Case Step Optimization Check Example

State Number, j	Step Size, m	Violation Matrix	Step Check, F(X')-F(X) (km/s)	Check Sum
1	1.00E-07	0 0 0 0 0 0 1	7.47E-12	1
2	1.00E-07	0 0 0 0 0 0 0	2.93E-11	0
3	1.00E-07	0 0 0 0 0 0 0	3.02E-11	0
4	1.00E-07	0 0 0 0 0 0 0	3.07E-11	0
5	1.00E-07	0 0 0 0 0 0 0	-7.93E-12	0
6	1.00E-07	0 0 0 0 0 0 0	-1.93E-11	0
7	1.00E-07	0 0 0 0 0 0 0	-1.98E-11	0
8	1.00E-07	0 0 0 0 0 0 0	-1.79E-11	0
1	-1.00E-07	0 0 0 0 0 0 0	-7.47E-12	0
2	-1.00E-07	0 0 0 0 0 0 0	-2.93E-11	0
3	-1.00E-07	0 0 0 0 0 0 0	-3.02E-11	0
4	-1.00E-07	0 0 0 0 0 0 0	-3.07E-11	0
5	-1.00E-07	0 0 0 0 0 0 0	7.93E-12	0
6	-1.00E-07	0 0 0 0 0 0 0	1.93E-11	0
7	-1.00E-07	0 0 0 0 0 0 0	1.98E-11	0
8	-1.00E-07	0 0 0 0 0 0 0	1.79E-11	0
1	1.00E-06	0 0 0 0 0 0 1	**7.47E-11**	**1**
2	1.00E-06	0 0 0 0 0 0 0	**2.93E-10**	**0**
3	1.00E-06	0 0 0 0 0 0 0	**3.02E-10**	**0**
4	1.00E-06	0 0 0 0 0 0 0	**3.07E-10**	**0**
5	1.00E-06	0 0 0 0 0 0 1	**-7.93E-11**	**1**
6	1.00E-06	0 0 0 0 0 0 1	**-1.93E-10**	**1**
7	1.00E-06	0 0 0 0 0 0 1	**-1.98E-10**	**1**
8	1.00E-06	0 0 0 0 0 0 1	**-1.79E-10**	**1**
1	-1.00E-06	0 0 0 0 0 0 1	**-7.47E-11**	**1**
2	-1.00E-06	0 0 0 0 0 0 1	**-2.93E-10**	**1**
3	-1.00E-06	0 0 0 0 0 0 1	**-3.02E-10**	**1**
4	-1.00E-06	0 0 0 0 0 0 1	**-3.07E-10**	**1**
5	-1.00E-06	0 0 0 0 0 0 1	**7.94E-11**	**1**
6	-1.00E-06	0 0 0 0 0 0 0	**1.93E-10**	**0**
7	-1.00E-06	0 0 0 0 0 0 0	**1.98E-10**	**0**
8	-1.00E-06	0 0 0 0 0 0 0	**1.79E-10**	**0**

Unreasonable but Feasible Solutions.

Some of the optimization runs produced unreasonable circumnavigations, but still met the mathematical constraints. For instance, placing an initial guess with only two burn points spaced by π radians for the same circumnavigation requirements used in Figure 8, yields an optimized path that doesn't circumnavigate the chief, but still lies in

the feasible space of the cost function. The initial guess is itself infeasible, but the resulting feasible optimized path is shown in Figure 11. The optimized Δv_t is 0.84460 m/s. Although the optimized path does not actually circumnavigate the chief, it does meet the constraints as stated above. This type of minimum was only found when the initial guess is infeasible.

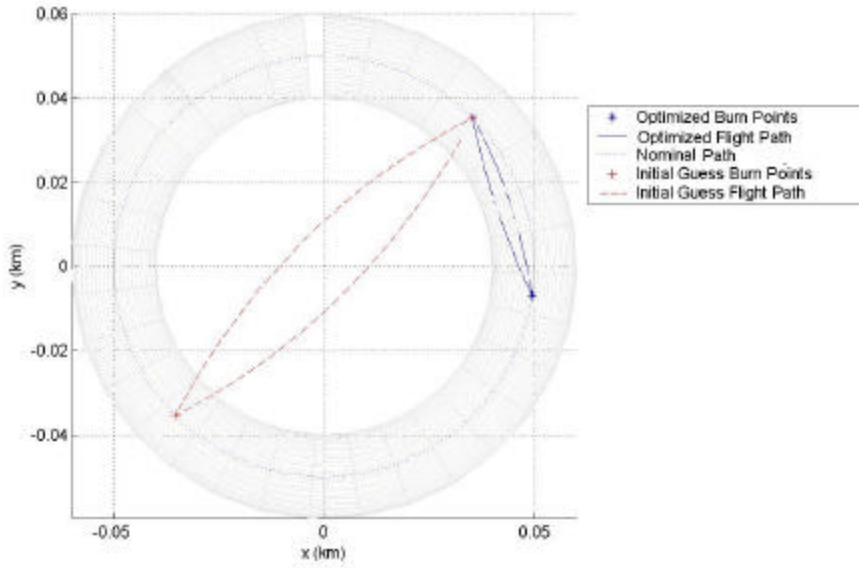


Figure 11. Two Burn Points. $\Theta y=90$, $\Theta z=0$, $\gamma o=45$ with TOF = 0.1 and $\rho max=0.01$ km

As the initial conditions are modified by placing the initial point ahead or behind the chief along the y-axis, ($\gamma_o = 0$ or π), a minimum can be found by moving infinitesimally 'forward' along the path, and infinitesimally 'back' to the initial position. The Δv_t for these 'circumnavigations' approach zero, which represents the global minimum for those conditions. Of course, these cases do not represent valid circumnavigations. This can be avoided by ensuring an adequate number of burn points

are used in the initial guess, that they are sufficiently spaced apart, and the initial guess is feasible.

Special Case Results Evaluation.

Several other nominal paths, TOF requirements, and constraint boundaries were investigated using the special case shown in Table **3**. The EAET method was used as the initial guess for all of the outputs.

Table 3. Special Case Representative Results

	Initial Conditions r_0 **= 50 m, a = 6778 km**	**Total Impulse (**Δv_t**)**		
		EAET Δv_t		**Special Case Optimization Results**
		Min. Burn Pts. (b)	Δv_t **(m/s)**	Δv_t **(m/s)**
				% of EAET
Variation of Nominal Path Orientation	Q_y**=60°,** Q_z**=30°,** g_o**=45°, TOF = 0.1,** r_{max}**=10 m**	5	2.67	2.55
				4.3%
	Q_y**=0°,** Q_z**=0°,** g_o**=0°, TOF = 0.1,** r_{max}**=10 m**	5	3.02	2.96
				2.2%
	Q_y**=90°,** Q_z**=0°,** g_o**=45°, TOF = 0.1,** r_{max}**=10 m**	5	2.61	2.49
				4.5%
Variation of r_{max}	Q_y**=60°,** Q_z**=30°,** g_o**=45°, TOF = 0.1,** r_{max}**=20 m**	4	2.39	2.11
				11.8%
	Q_y**=60°,** Q_z**=30°,** g_o**=45°, TOF = 0.1,** r_{max}**=8 m**	6	2.85	2.70
				5.1%
Variation of TOF	Q_y**=60°,** Q_z**=30°,** g_o**=45°, TOF = 0.2,** r_{max}**=10 m**	5	1.18	1.13
				4.4%
	Q_y**=60°,** Q_z**=30°,** g_o**=45°, TOF = 0.05,** r_{max}**=10 m**	5	5.67	5.49
				3.2%

The optimization results for varying the orientation of the nominal path shows the limit for how well the optimization performs over the EAET case. The best the optimization performed was when the nominal path is in the x-y plane, as expected for a

constant deviation constraint. Conversely the smallest difference between the EAET and the optimization occurred when the nominal path was not rotated at all, which corresponds to the y-z plane circumnavigation. These results are consistent with Figures 6 and 7 as well as the continuous case results (4:3-8). The variation show in the y-z plane nominal path gives credibility to extrapolating the effects of nominal path orientation variation as well as TOF variation.

Comparing the total impulse value for differing ρ_{max} constraints, while keeping the circumnavigation path and TOF constant, shows that as ρ_{max} decreases, the Δv_t increases. This result is consistent with the results implied from the EAET analytical method; the minimum number of burn points has to increase to meet the flight path constraint and as the minimum number of burn points increases the Δv_t increases as well. Another expected result is the Δv_t scales directly with the TOF; shorter TOF's result in greater Δv_t.

General Case Results

The General Case allows the burn points to vary off of the nominal path with three spatial degrees of freedom, but all other constraints apply. Figure 12 shows the results with $\Theta_y = 90^o$, $\Theta_z = 0^o$, $\gamma_o = 45^o$, TOF = 0.1, and ρ_{max}=0.01 km, and the minimum EAET, b=5, as the initial guess (equivalent to Figure 8). The optimization routine found a local minimum with the intermediate burn points placed on the nominal path. The optimized Δv_t doubles the savings from the optimized special case at 2.3754 m/s representing 8.92% savings. The flight path touches the inner constraint radius at four points with the optimization routine minimizing the path length between the third and

fourth burn points. While all the burn points still lie on the nominal path, the endpoint is now located on the outside constraint boundary.

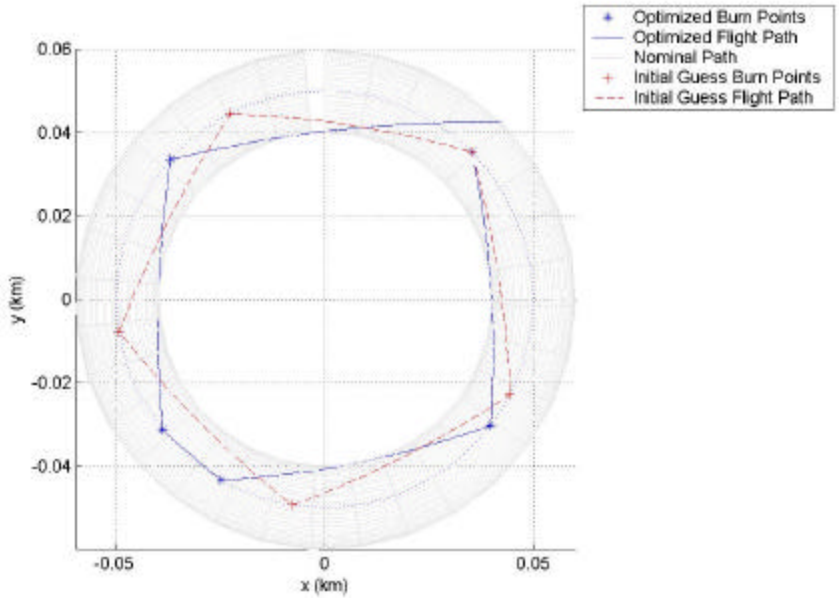


Figure 12. General Case Optimization Results for Θ_y=90, Θ_z=0, γ_o=45, TOF = 0.1 and ρ_{max}=0.01 km

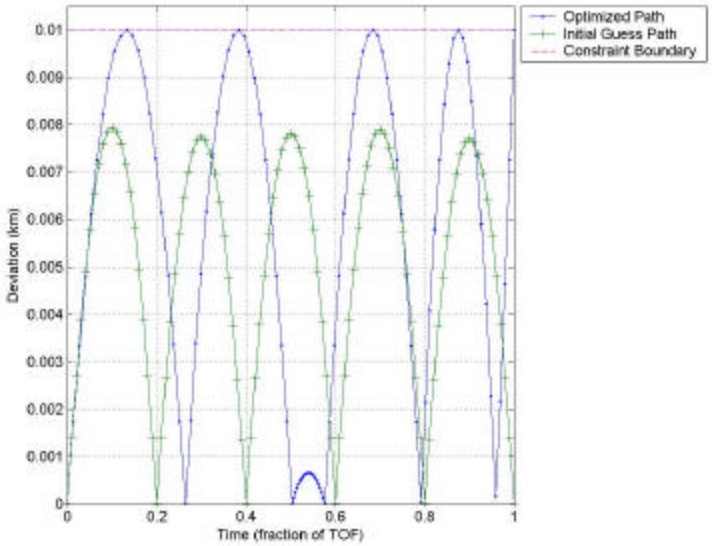


Figure 13. General Case Optimization Results Deviation for Θ_y=90, Θ_z=0, γ_o=45, TOF = 0.1 and ρ_{max}=0.01 km

Figure 12 shows an interesting result; all burn points still lie on the nominal path when they are free to vary off of it. This indicates the cost function has a local minimum when the burn points are placed on the nominal path and corresponds to the special case above.

Varying Initial Guess Radii.

The numerical results are sensitive to different initial guesses. The next approach is to run the optimization placing the initial guess burn points away from the nominal path. The guesses are defined by using the EAET placement in γ and t along a varying radius away from the nominal path radius, but within the constraint boundary. Figure 14 shows the placement of the initial guess burn points on the extreme outer constraint boundary, at a radius of 60 meters. The end point of the initial guess is set on the outside edge of the constraint boundary. The optimization routine finds a local minimum very close to the EAET guess placed on the nominal radius with the burn points placed within 1-3 meters off the nominal radius.

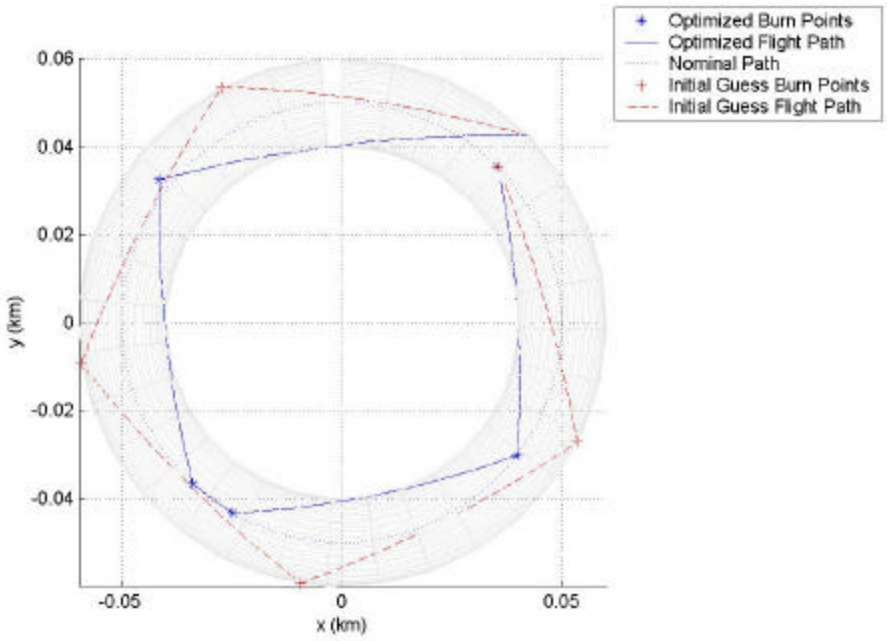


Figure 14. General Case Optimization with Guess on Outer Constraint Boundary. Θ_y=90º, Θ_z=0º, γ_o=45º with TOF = 0.1 and ρ_{max}=0.01 km

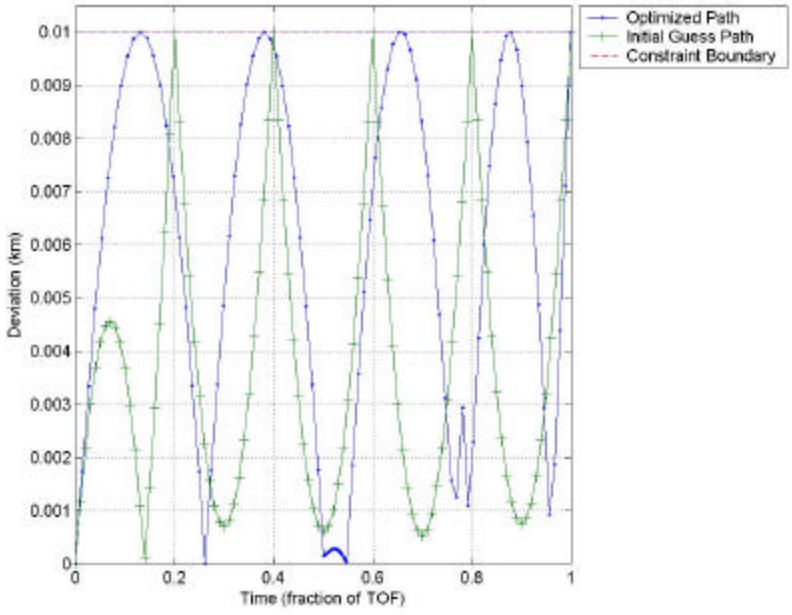


Figure 15. Deviation of Guess on Outer Constraint Boundary. Θ_y=90º, Θ_z=0º, γ_o=45º with TOF = 0.1 and ρ_{max}=0.01 km

Figure 15 shows the deviation of the initial guess on the outside constraint boundary and the optimization. Again the optimization finds the state vector such that the intermediate flight paths skirt the inner constraint boundary, while trying to eliminate a burn point by minimizing the path between the third and fourth burn points. The Δv_t for this optimization is 2.3775 m/s or 8.85 % savings on EAET which is approximately the same as the results for the initial guess on the nominal path.

Placing the initial guess burn points on the interior radius of the constraint surface leads to non-convergence in the optimization. The TOF of 0.1 causes every flight path point except for the burn points to be in the cost function's infeasible region. No fast TOF was found to converge on a solution if all burn points were placed on the inner constraint radius.

The next step is to progressively step the initial guess radius (constant radius upon which the EAET points are located) inward (toward the inner constraint radius) from r_0. Figure 14 shows the results when the 5 point EAET guess is placed upon a radius of 48.5 m (1.5 m less than the nominal of 50 m). The initial guess Δvt is found to be 2.5324 m/s and the optimized Δvt is 2.378 m/s. The optimized solution represents 8.9408% savings on the EAET method at the nominal radius. The key feature of the new optimized solution is the fact that all intermediate paths now skirt the inner constraint radius.

The Δv_t is approximately the same as the previous two cases, indicating a region in which the objective function is 'flat'; resulting in a numerically sensitive search that yields many different local minima. These local minima are only stationary points due to numerical imprecision.

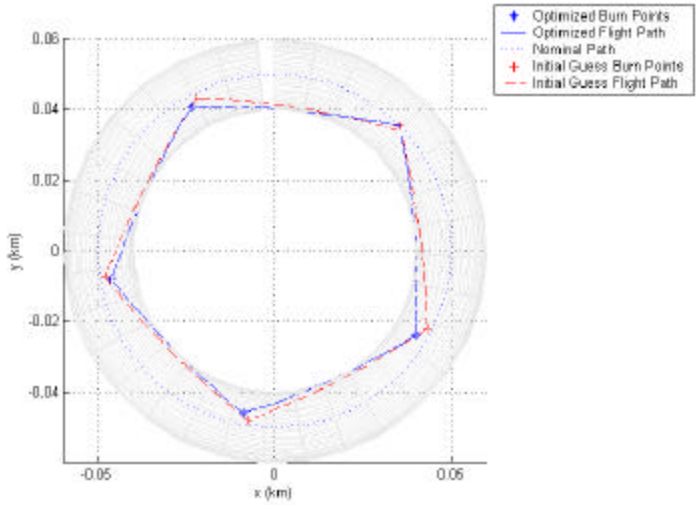


Figure 16. Guess Radius of 48.5 m. $\Theta_y = 90^o$, $\Theta_z = 0^o$, $\gamma_o = 45^o$ with TOF = 0.1 and $\rho_{max} = 0.01$ km

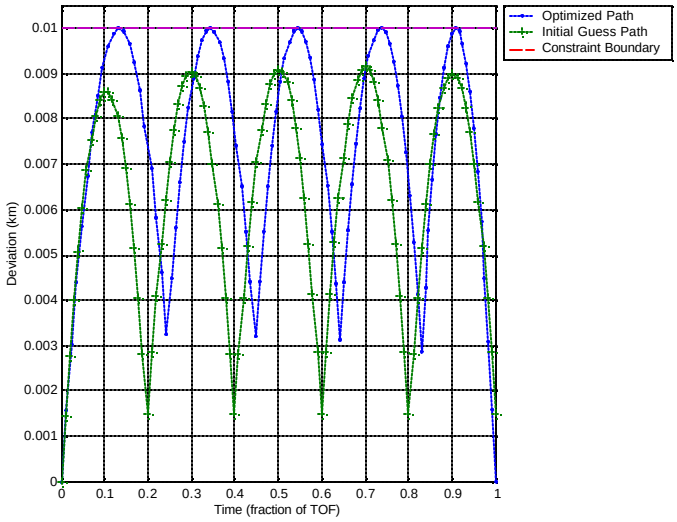


Figure 17. Guess Radius of 48.5 m, Deviation. $\Theta_y = 90^o$, $\Theta_z = 0^o$, $\gamma_o = 45^o$ with TOF = 0.1 and $\rho_{max} = 0.01$ km

Placing the 5 burn point initial guess on a radius tighter than the average radius of 3 m shown in Figure 17 produces an infeasible guess that will not converge to a solution for these parameters; therefore the number of burn points must be increased in order to find feasible initial guesses that will converge to a solution.

Figure 18 represents the guess with an inner radius one third of the distance from the nominal radius to the inner constraint boundary while increasing *b* to six. The optimization finds a minimum touching the inner constraint boundary between every burn point. The resulting optimized intermediate burn points are placed at a nearly constant ρ_{dev} of 6 meters as seen in the deviation of the optimization in Figure 19.

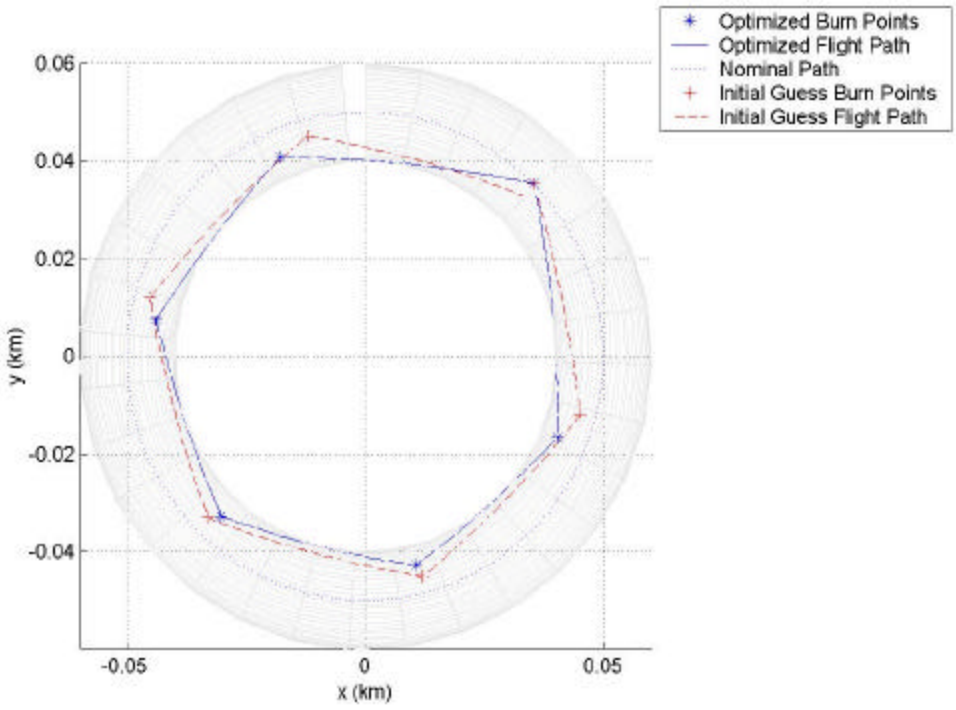


Figure 18. Initial Guess 1/3 Less Nominal Radius and *b* = 6. Θ_y=90º, Θ_z=0º, γ_o=45º with TOF = 0.1 and ρ_{max}=0.01 km

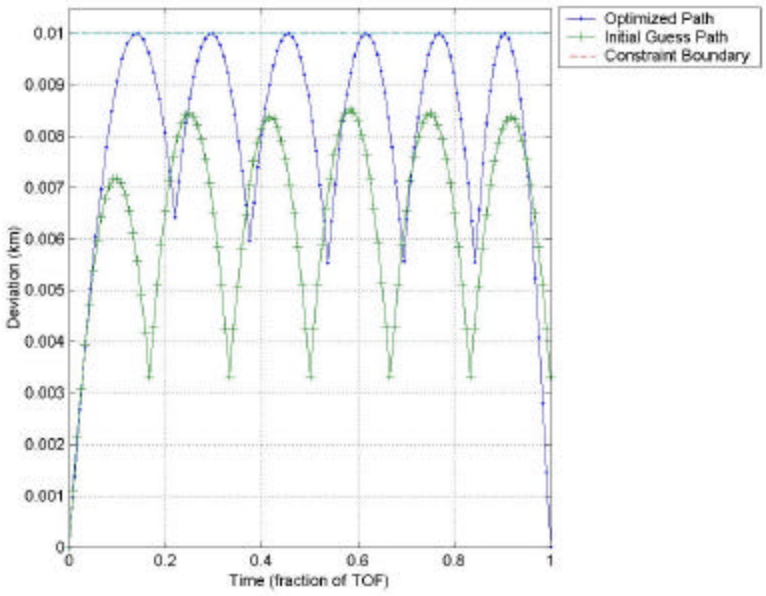


Figure 19 Deviation with Initial Guess 1/3 Less Nominal Radius and b = 6. Θ_y=90º, Θ_z=0º, γ_o=45º with TOF = 0.1 and ρ_{max}=0.01 km

The one third radius guess's Δv_t is 2.5923 m/s; whereas the optimized Δv_t is 2.3360 m/s. This represents a 0.61% savings and 10.44% savings respectively compared to the minimum EAET at the nominal radius. The increase in savings from the optimized solution shown in Figure 16, is counter to the EAET conclusion of the lower the number of burn points the lower the Δv_t.

The initial guess radius is tightened to 6.5 meters while keeping the number of burn points constant at six. Figures 20 and 21 illustrate the results from this initial guess. Note the initial guess is in the cost function's infeasible region, but the optimization routine is able to converge on a minimum. The savings from the initial guess (compared to EAET) is 7.23% at a value of 2.4197 m/s, whereas the savings from the optimization is 10.42% at a value of 2.3365 m/s. This shows there is a unique radius for a given number of burn points where all the intermediate flight paths are tangential to the inner constraint radius. This fact will be used to develop an analytical method for determining the placement of the burn points.

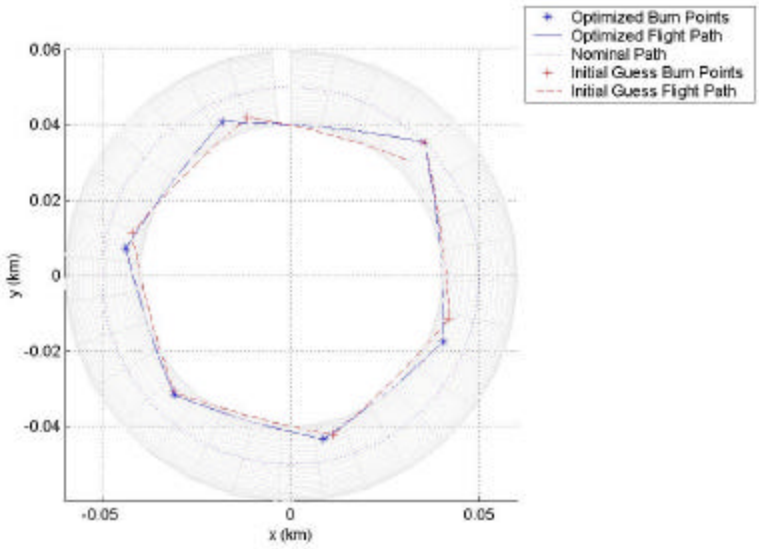


Figure 20. Initial Guess Radius = 6.5 m and b = 6. Θ_y=90º, Θ_z=0º, γ_o=45º with TOF = 0.1 and ρ_{max}=0.01 km

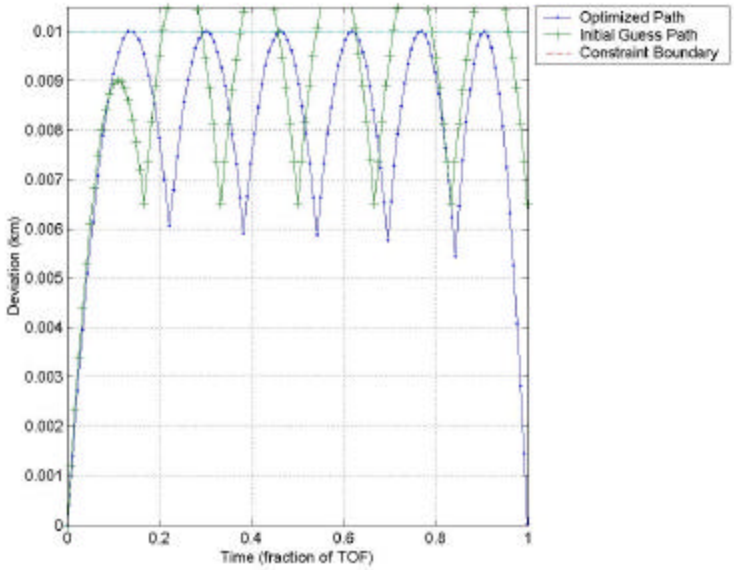


Figure 21. Deviation of Initial Guess Radius = 6.5 m and b = 6. Θ_y=90º, Θ_z=0º, γ_o=45º with TOF = 0.1 and ρ_{max}=0.01 km

Out of Nominal Plane Component.

Most of the burn points do not have a significant out-of-nominal-plane placement for most rotations. The maximum out of nominal plane placement of the burn points occurs when the nominal path is not rotated, but left in the y-z plane. Figure 22 shows an edge on view of a circumnavigation with r0 = 50 m, $\Theta_y = 0^o$, $\Theta_z = 0^o$, $\gamma_o = 45^o$, and a TOF = 0.1. The maximum deviation radius is set at 10 m, but the maximum deviation out of plane is 2 m. The initial guess is the EAET with a radius one third less than the nominal (the same as presented in Figure 22). The small deviation is used as a simplifying assumption in the development of an analytical method below.

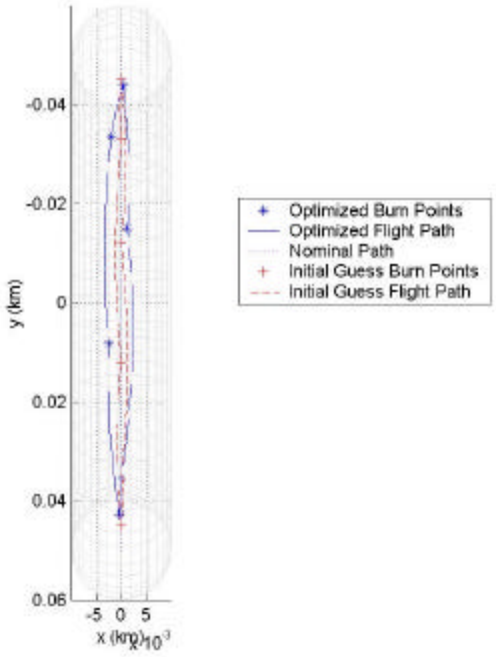


Figure 22. y-z Plane Circumnavigation

General Case Results Evaluation.

The main trend from the general case optimization is the intermediate flight paths all skirt the inner constraint radius. These circumnavigations represent the least Δv_t

found for a given number of burn points, and the shortest path lengths between the burn points.

There are two additional trends. First, given a set of rotation angles, constraint boundary, and an initial guess with a tighter radius the number of burn points must increase to find a feasible solution. However, this increase in the number of burn points does not increase Δv_t as indicated from the EAET method. Second, there appears to be radius which allows a balance between the dynamics of the system and shortening the path length for a given number of burn points. Additionally, the resulting out-of-nominal-plane component for optimized burn points is small.

The initial guesses have so far produced hints as to where the minima are located, but a new method for developing an initial guess is desired. A better guess will allow for a more comprehensive search for the best circumnavigation maneuver.

V. Analytical Design Method

Based on the general case optimization results, an improved analytical design method is developed yielding a lower Δv_t than the EAET method. For fast TOF (less than 0.3 of the chief's orbital period), it is hypothesized that a minimal total length path that skirts the inner constraint radius is optimal for a given number of burn points. The path can be approximated as straight lines tangential to the inner constraint radius. Furthermore, there exists a unique radius from the chief upon which the burn points are placed. This radius (termed the design radius, r_d) is no more than the nominal radius and no less than the nominal radius minus the maximum deviation radius (the inner constraint radius). Trends in the general case results justify this hypothesis.

The general case results justify this hypothesis based upon three factors. First, the general case optimization results indicate minima with all the intermediate flight points touching the inner constraint radius. Next, the burn points lie within a nearly constant radius greater than the inner constraint radius, but less than the nominal radius. Finally, the optimized burn points lie within a plane very near the plane of the nominal path.

Analytical Design Assumptions and Simplifications

In creation of the design algorithm, several factors are assumed and simplifications made from the general case optimization results. The flight paths are assumed to be straight lines. Additionally, the algorithm places the 'designed' plane to coincide with the nominal path's plane. The times of flight between burn points are assumed to be the same ratio with TOF as the burn point's γ angle from the last burn point to 2π. The end point for the circumnavigation is constrained to lie on the outside constraint boundary on a line between initial point and the chief. Finally, the γ angles

between the 2nd burn point and the next to last burn point are assumed to be equally divided.

Algorithm Description and Analysis

All burn points but the initial burn point lie on a circle with the design radius, r_d. The inner constraint radius, r_c, is computed by subtracting ρ_{max} from r_0. A search range for the values of the design radius, r_d, is a set of the radii from the inner constraint radius to the nominal radius. A set of burn points is determined for each potential design radius within this range. Figure 23 shows the corresponding position and relevant angles for all the burn points for a unique design radius.

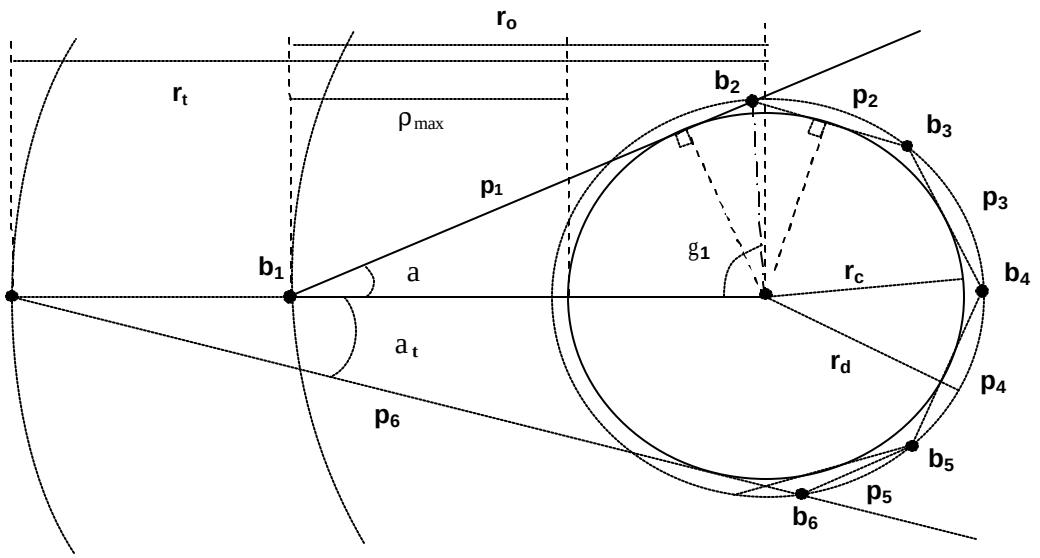


Figure 23 Sketch of Design Algorithm Geometry

The second burn point, b_2, is determined by finding the tangent line from the initial point, b_1, to the inner constraint radius in the circumnavigation's direction, and placing b_2 at the farthest intersection of the tangent line and the circle inscribed by r_d. In order to compute the γ angle between the two burn points, the angle α must be found by

$$a = \sin^{-1}\left(\frac{r_c}{r_o}\right) \tag{31}$$

The initial path length, p_1, between the burn points is computed using the Pythagorean Theorem:

$$p_1 = \sqrt{r_o^2 - r_c^2} + \sqrt{r_d^2 - r_c^2} \tag{32}$$

Using this path length and the angle, α, γ_1 is computed using the Law of Sines:

$$g_1 = \sin^{-1}\left(\frac{p_1 \sin(a)}{r_d}\right) \tag{33}$$

The next step is to compute the last burn point (in Figure 23 it is b_6). The angle α_t is computed exactly as Eq. (31), but r_0 is replaced with the radius to the end point, r_t. Eq. (32) is used again to determine the final path length, p_b, however r_0 is replaced by r_t in Eqs. (31) and (32). Next, Eq. (33) is used to determine the last angle, γ_b, replacing the appropriate variables.

Once the second and last burn points have been determined, the intermediate burn points are calculated. A tangent line to the inner constraint radius is circled from the second burn point and each subsequent burn points. The next burn point is placed at the intersection of the tangent line and the design radius. The γ value for each burn point is determined by simple geometry. This procedure is accomplished until the sum of previous γ values exceeds $2\pi - \gamma_{b-1}$. The next to the last angle, γ_{b-1}, (in Figure 23: γ_5) is calculated as

$$g_{b-1} = 2p - \left(\sum_{i=1}^{b-2} g_i + g_b\right) \tag{34}$$

The next step is to iterate the cost function through each value of the design radii. This iteration can be visualized in Figure 24. Figure illustrates four design radii, special attention should be drawn to the two middle circles (specifically to the drawn area of interest) and the next-to-the-last path length. For the greater of the two circles, the next-to-the-last path is tangent to the inner constraint radius. However, decreasing the design radius to the next circle produces a next-to-the-last path that is not tangent to the inner constraint radius. There exist specific design radii which produce a flight path tangent to the inner radius between each burn point including the next to the last flight path.

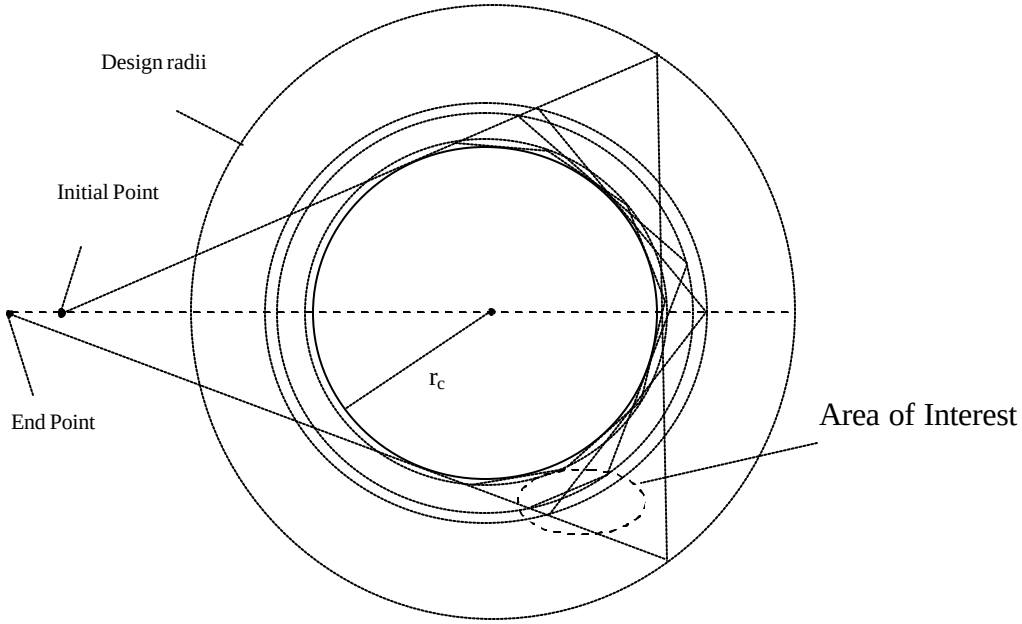


Figure 24 Sketch of Straight Line Flight Paths with Multiple Design Radii

Now that the burn points are placed, the total circumnavigation velocity change, Δv_t, is calculated for a range of design radii. The design radius producing the minimum Δv_t is then chosen. Figure 25 shows the Δv_t as a function of r_d used to determine the final design radius.

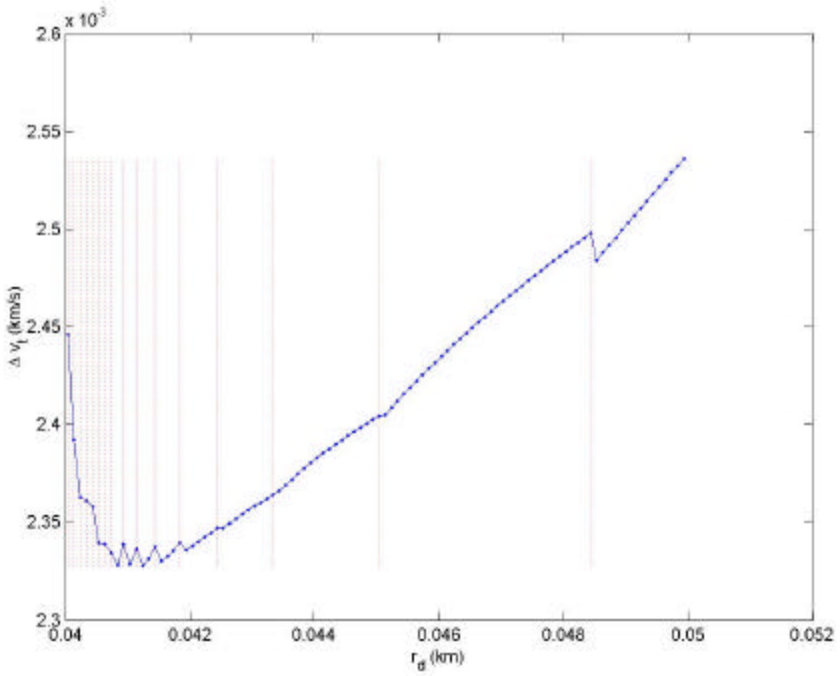


Figure 25. Δv_t vs r_d. $\Theta_y = 90^o$, $\Theta_z = 0^o$, $\gamma_o = 45^o$ with TOF = 0.1 and $\rho_{max} = 0.01$ km

The minimum for this case occurs at a r_d of 0.04124 km. The vertical dashed lines represent transitions from one discrete number of burn points to another in the analytical design. The jumps in Δv_t in Figure 25 are directly correlated to these transitions. The reason for this behavior is seen in Figure 24 where the p_{b-1} path is only tangent at unique values of the design radius as discussed above.

An example of the MATLAB code for this algorithm is located in Appendix B.

Variation of Number of Burn Points with Design Radius.

The number of burn points grows exponentially as the design radius approaches the inner constraint radius, which can be seen in Figure 26.

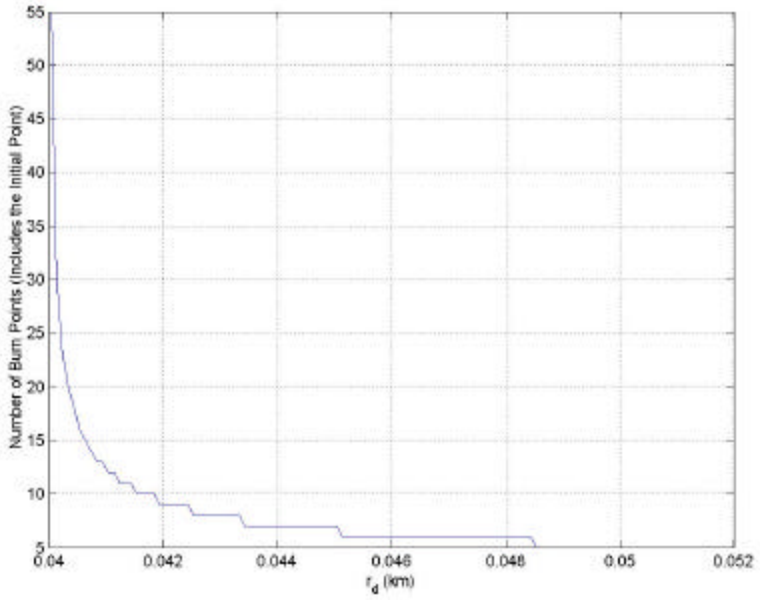


Figure 26. Number of Burn Points vs. $r_{d.}$ $\Theta_y = 90^o$, $\Theta_z = 0^o$, $\gamma_o = 45^o$ with TOF = 0.1 and $\rho_{max} = 0.01$ km

The minimum Δv_t does not correlate to the minimum number of burn points. This result seemingly contradicts the conclusion above that the lower the number of burn points the lower Δv_t. The number of burn points is increased in order to shorten the flight path to skirt the inner constraint radius. Eventually, though, increasing the number of burn points (tightening the design radius) no longer decreases the Δv_t but increases it.

End Point Position Selection.

The decision to nominally place the end point on the outer constraint boundary was determined empirically by investigating many general case solutions, and experimentally by varying the end point for a specific circumnavigation maneuvers analytical design. The analytical design algorithm is modified to change the position of the end point for the maneuver. This is accomplished by varying the value of r_t from the inner torus radius to the outer torus radius and computing the total impulse for each

design. Figure 27 shows the impact on the total impulse with varying the radius to the end point for the circumnavigation maneuver defined by the parameters: $\Theta_y = 90^o$, $\Theta_z = 0^o$, $\gamma_o = 45^o$ with TOF = 0.1 and $\rho_{max} = 0.01$ km. The maximum value for Δv_t occurs on the inner constraint radius, and the minimum value occurs at the outer constraint radius.

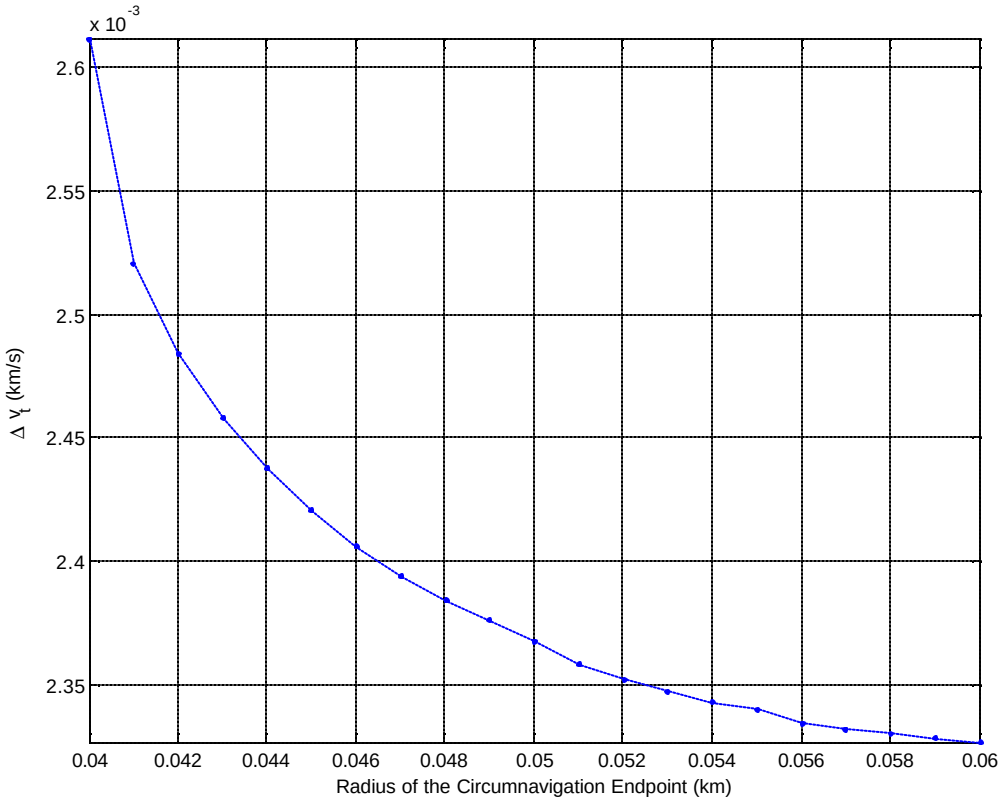


Figure 27. End Point Variation in the Analytical Design Method, $\rho_{max} = 0.01$ km.

There are circumnavigations for which placing the endpoint on the outer constraint boundary does not represent the minimum analytical design choice. Figure 28 shows the results of varying the endpoint for the same circumnavigation, but with an increased maximum deviation radius: $\rho_{max} = 0.02$ km.

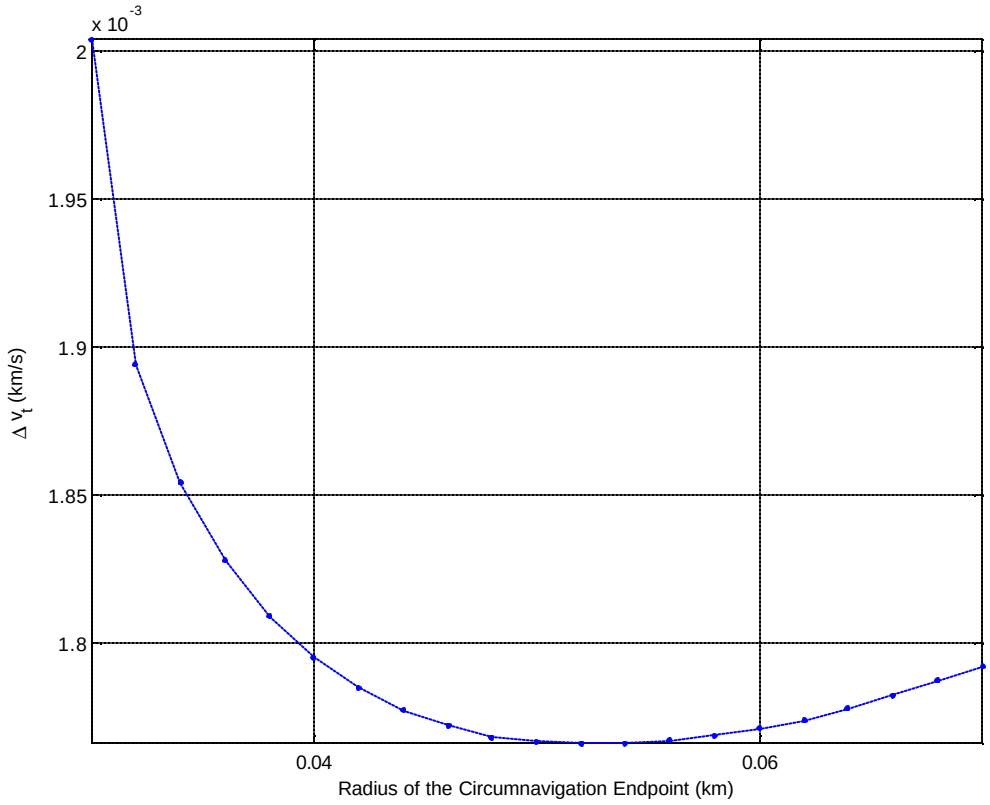


Figure 28. End Point Variation in the Analytical Design Method, ρ_{max} = 0.02 km.

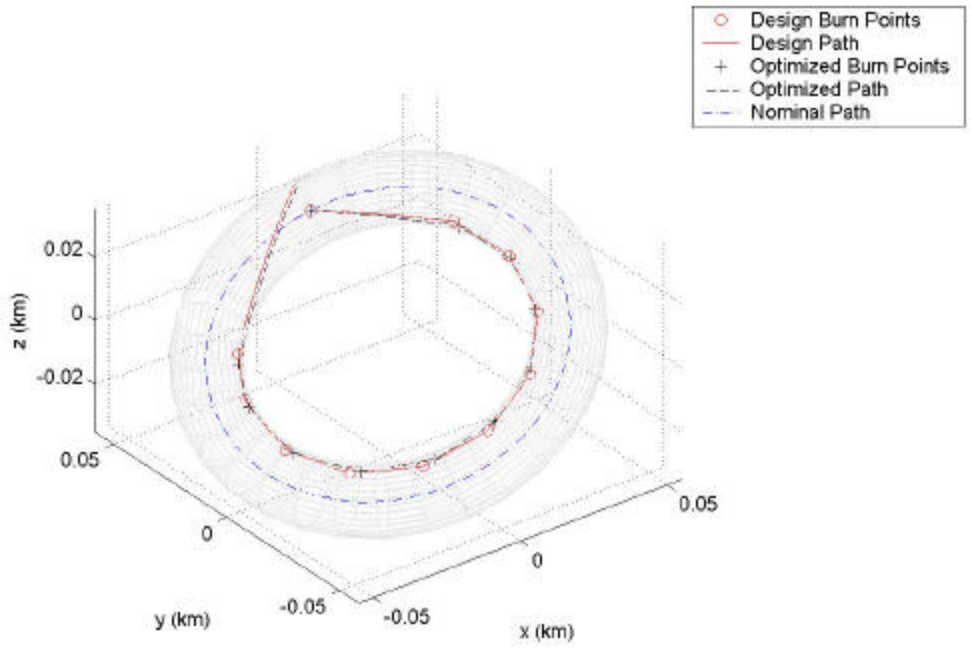
This shows the optimal placement of the endpoint near the nominal radius; however the difference between placing the endpoint on its optimal position and the outer constraint boundary is not great. Additionally, the large increase in the maximum deviation radius for this circumnavigation may not be realistic for operational constraints. To simplify the algorithm, the end point is assumed to lie on the outer constraint boundary for the rest of the results presented. The optimal end point placement can be used if the maximum deviation radius becomes a significant fraction of the nominal radius, however for all examples in this thesis the end point has been placed on the outer constraint boundary.

Analytical Design Method Performance

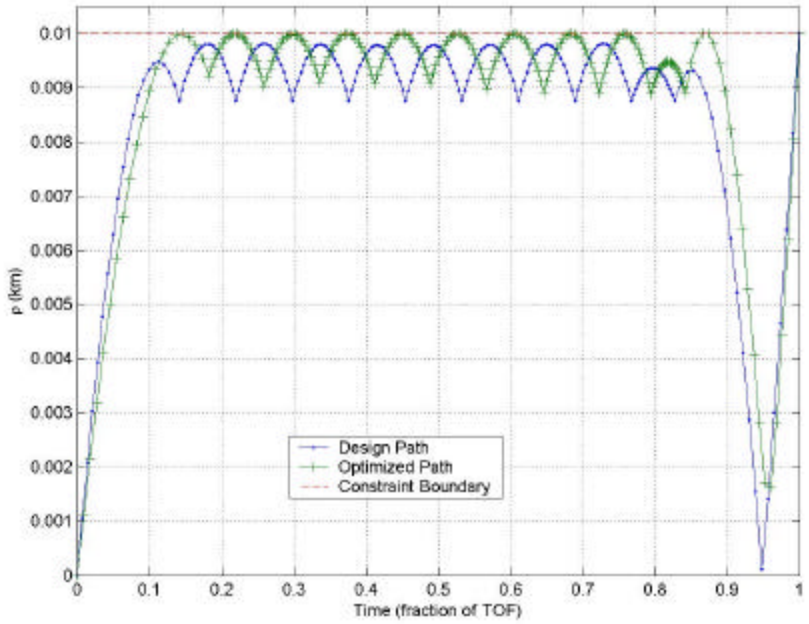
The designed algorithm for the initial conditions is input as the initial guess into the optimization routine. This allows for significant savings compared to the EAET case, but also validates the use of the algorithm as a good approximation near a local minimum.

Optimization Results with Design as Initial Guess.

Figure 29 shows the optimization results using the design algorithm guess for r_0 = 50 m, $\Theta_y = 60^o$, $\Theta_z = 30^o$, $\gamma_o = 45^o$ with TOF = 0.1 and ρ_{max}=0.01 km. Figure 30 shows the deviation of both the design and the optimized solution.



Figures 29. Design and Optimized Circumnavigation Paths. r_0 = 50 m, $\Theta_y = 60^o$, $\Theta_z = 30^o$, $\gamma_o = 45^o$ with TOF = 0.1 and ρ_{max} = 0.01 km



Figures 30. Design and Optimized Circumnavigation Path Deviations. r_0 = 50 m, Θ_y = 60º, Θ_z = 30º, γ_o = 45º with TOF = 0.1 and ρ_{max} = 0.01 km

The two paths are very close to each other, with the optimized path touching the inner constraint radius at most points. The design actual path does not actually touch the constraint radius because the actual flight paths between burn points are curved as opposed to straight lines. The Δv_t for the design path is 2.3928 m/s with the optimized Δv_t equal to 2.2942 m/s which represent savings of 10.30% and 14.00% respectively. Savings realized by using the optimized state vector as opposed to the design state vector are only 0.0985 m/s, or 3% on EAET. The computational requirements for the optimization may not justify this small increase.

Variation of the Inner Constraint Radius (Maximum Deviation Radius).

As seen in the Special Case results varying the maximum deviation radius has an impact on the placement of the burn points for the optimized solution. For the analytic method the number of burn points increases as expected (subsequently the Δv_t increases)

as the maximum deviation radius decreases. The inverse relationship is true as well. Figures 31 and 32 show the circumnavigation paths and deviations respectively for the same circumnavigation shown in Figure 29, but with ρ_{max} increased to 20 m. The Δv_t for the design path is 1.84 m/s with the optimized Δv_t equal to 1.72 m/s which represent savings of 21.25% and 28.33% respectively over an EAET method using 4 burn points and a Δv_t = 2.40 m/s.

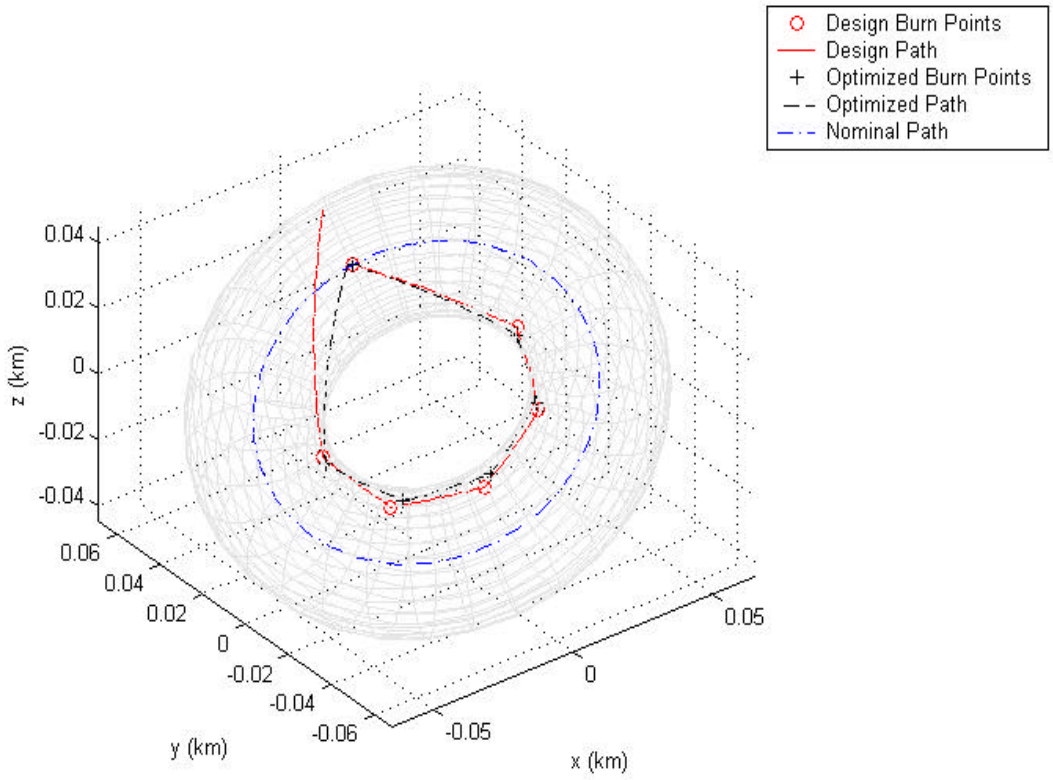


Figure 31. Design and Optimized Circumnavigation Paths—Larger ρ_{max}. r_0 = 50 m, Θ_y = 60°, Θ_z = 30°, γ_o = 45° with TOF = 0.1 and ρ_{max} = 0.02 km

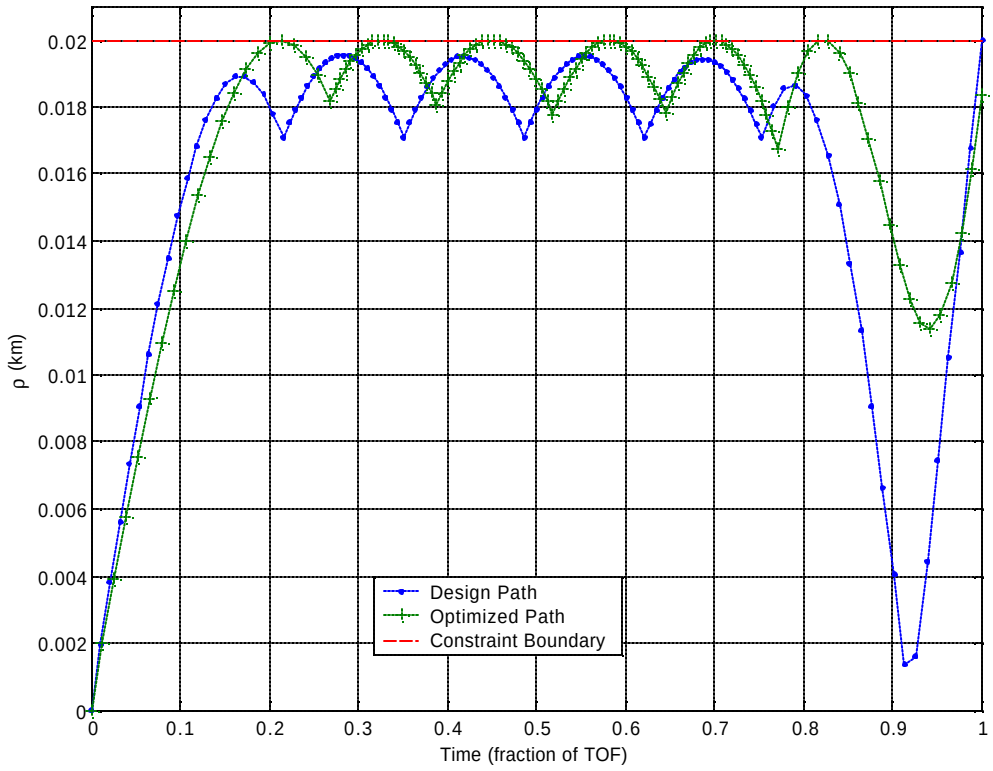
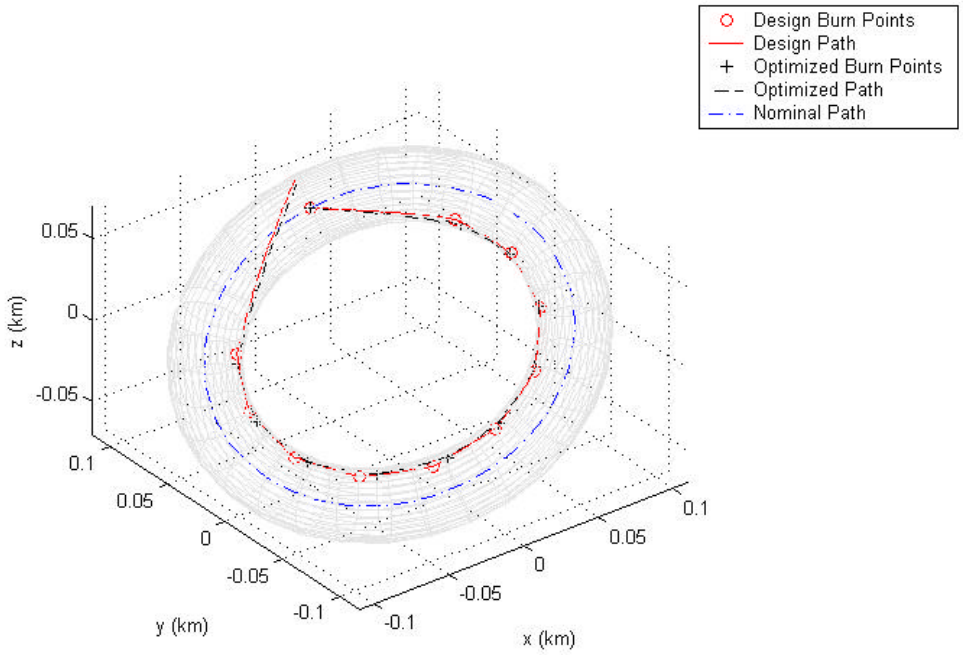


Figure 32. Design and Optimized Circumnavigation Deviations—Larger ρ_{max}. r_0 = 50 m, $\Theta_y = 60^o$, $\Theta_z = 30^o$, $\gamma_o = 45^o$ with TOF = 0.1 and ρ_{max} = 0.02 km

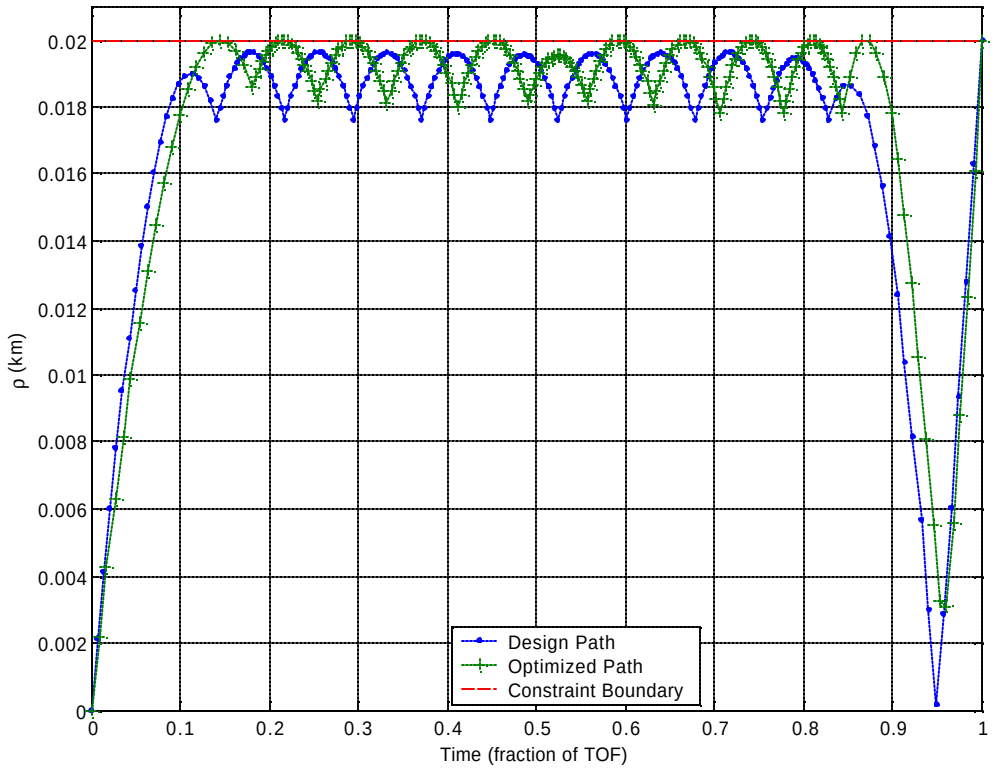
The design radius for this case is farther from the inner constraint radius at 32.9 m (2.9 m away from the inner constraint radius). The major increase in savings is a result of the total path length for the circumnavigation being considerably shorter (tighter radius) than the EAET method, even with reducing the number of burn points. The amount of decrease in Δv_t for the EAET due to the decrease in the number of burn points (compared to the previous one) does not compensate for the better results found by the design and optimization. Overall, comparing the actual values of Δv_t, the general observation is made: increasing the ρ_{max} results in a decrease in Δv_t, consistent with all other cases.

Variation of the Nominal Radius.

Increasing the nominal radius shows the performance of the algorithm for another circumnavigation, and the percentage of savings over the EAET method stays relatively constant. Figures 33 and 34 show the results of increasing the nominal radius to 100m while making keeping the percentage (of the nominal radius) of the maximum deviation radius constant at 20% (20 m for 100 m). The orientation of the nominal path and TOF are the same as the circumnavigation shown in Figures 29 and 30 (r_0 = 100 m, $\Theta_y = 60^o$, $\Theta_z = 30^o$, $\gamma_o = 45^o$ with TOF = 0.1 and ρ_{max} = 0.02 km). The Δv_t for the design path is 4.78 m/s with the optimized Δv_t equal to 4.59 m/s which represent savings of 10.30% and 13.90% respectively.



Figures 33. Design and Optimized Circumnavigation Paths—Larger r_0.
r_0 = 100 m, $\Theta_y = 60^o$, $\Theta_z = 30^o$, $\gamma_o = 45^o$ with TOF = 0.1 and ρ_{max} = 0.02 km



Figures 34. Design and Optimized Circumnavigation Path Deviations— Larger r_0. r_0 = 100 m, $\Theta_y = 60^o$, $\Theta_z = 30^o$, $\gamma_o = 45^o$ with TOF = 0.1 and ρ_{max} = 0.02 km

There are two effects of note from this example. First, the percentage of savings from the EAET stays relatively constant. Keeping the same ratio of inner constraint radius to the nominal radius allows for relatively constant savings, which implies for any given orientation the results can be scaled at a particular TOF.

Second, the number of burn points stays relatively constant for both the EAET and the design/optimization. As the nominal radius is increased for a constant TOF, the time to travel between subsequent burn points decreases with increasing distance. Essentially, this indicates the paths between the burn points better approximate a line, and the

minimum number of points allowed with a given inner constraint radius stays nearly constant.

Circumnavigation Requirements Comparison.

The results from a representative set of initial conditions are shown in Table 4. The analytic design outperforms the minimum burn EAET case, and the optimization results using the analytical design method as the initial guess always produces the minimum circumnavigation Δv_t found from any guess. Another benefit to the analytical design as an initial guess is a very high probability of convergence to a minimum.

Table 4. Analytic Design Algorithm Performance

	Initial Conditions r_0 **= 50 m, a = 6778 km**	**Total Impulse (**Δv_t**)**					
		EAET Δv_t		**Analytical Design Algorithm**			
		Min. Burn Pts. (b)	Δv_t **(m/s)**	**Design Burn Points (b)**	r_d **(m)**	Δv_t **(m/s)**	**Optimization** Δv_t **(m/s)**
						% of EAET	**% of EAET**
Variation of Nominal Path Orientation	Q_y**=60°,** Q_z**=30°,** g_o**=45°, TOF = 0.1,** r_{max}**=10 m**	5	2.67	11	41.2	2.39	2.29
						10.3%	14.0%
	Q_y**=0°,** Q_z**=0°,** g_o**=0°, TOF = 0.1,** r_{max}**=10 m**	5	3.02	11	41.2	2.79	2.71
						7.8%	10.5%
	Q_y**=90°,** Q_z**=0°,** g_o**=45°, TOF = 0.1,** r_{max}**=10 m**	5	2.61	11	41.2	2.37	2.24
						10.8%	14.3%
Variation of r_{max}	Q_y**=60°,** Q_z**=30°,** g_o**=45°, TOF = 0.1,** r_{max}**=20 m**	4	2.39	13	32.9	1.84	1.73
						23.2%	27.8%
	Q_y**=60°,** Q_z**=30°,** g_o**=45°, TOF = 0.1,** r_{max}**=8 m**	6	2.84	13	42.9	2.54	2.44
						10.9%	14.1%
Variation of TOF	Q_y**=60°,** Q_z**=30°,** g_o**=45°, TOF = 0.2,** r_{max}**=10 m**	5	1.18	10	41.5	1.05	0.99
						11.3%	16.6%
	Q_y**=60°,** Q_z**=30°,** g_o**=45°, TOF = 0.05,** r_{max}**=10 m**	5	5.67	13	40.8	5.14	4.97
						9.4%	12.3%

Table 4 also presents other indications of the cost function's nature, specifically; there is a strong correlation between the savings and the inner constraint radius as well as the TOF. As Figure 35 shows, the percent savings between the EAET method and the Analytical Design Method decreases with increasing TOF. As the TOF increases the EAET becomes a better approximation. The local minimums produced using the design algorithm as the initial guess are the lowest, valid circumnavigations found of all the initial conditions investigated.

The discontinuous steps in the EAET curve represent changes in the minimum number of burn points. The EAET does not necessarily minimize the path length by having the intermediate flight path tangent to the inner constraint radius, which produces the major difference between the two curves. The analytical method has the advantage of producing a relatively smooth line in Figure 35 which eliminates inefficiencies created by the discretized nature of the EAET method.

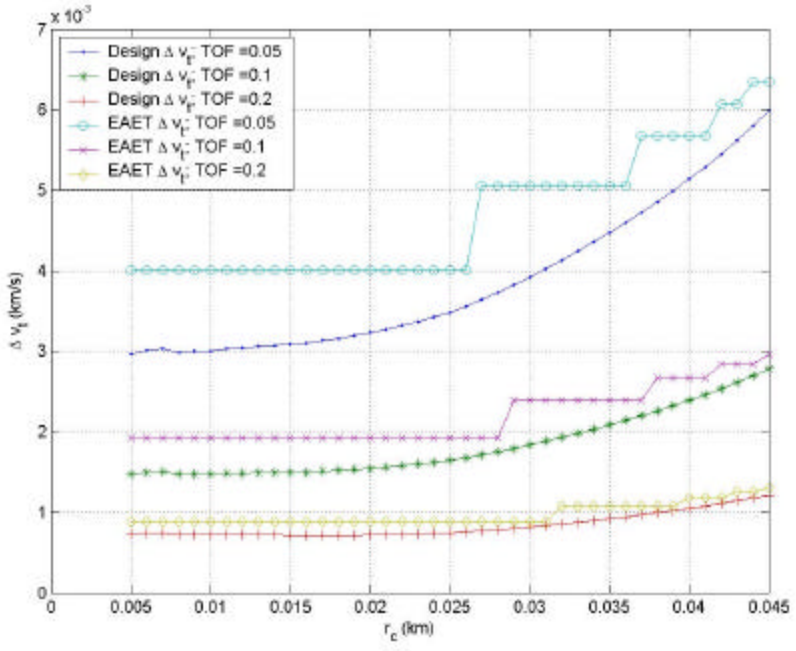


Figure 35. Δv_t versus r_c with varying TOF.
r_0 = 50 m, Θ_y=60°, Θ_z=30°, γ_o=45°, ρ_{max} = 0.01 km.

Design Radius Versus Inner Constraint Radius (Maximum Deviation Radius).

The analytical design method determines a design radius based upon the dynamics and the inner constraint radius. It is informative to investigate how the design radius varies as a function of the inner constraint radius for a particular circumnavigation which is presented in Figure 36.

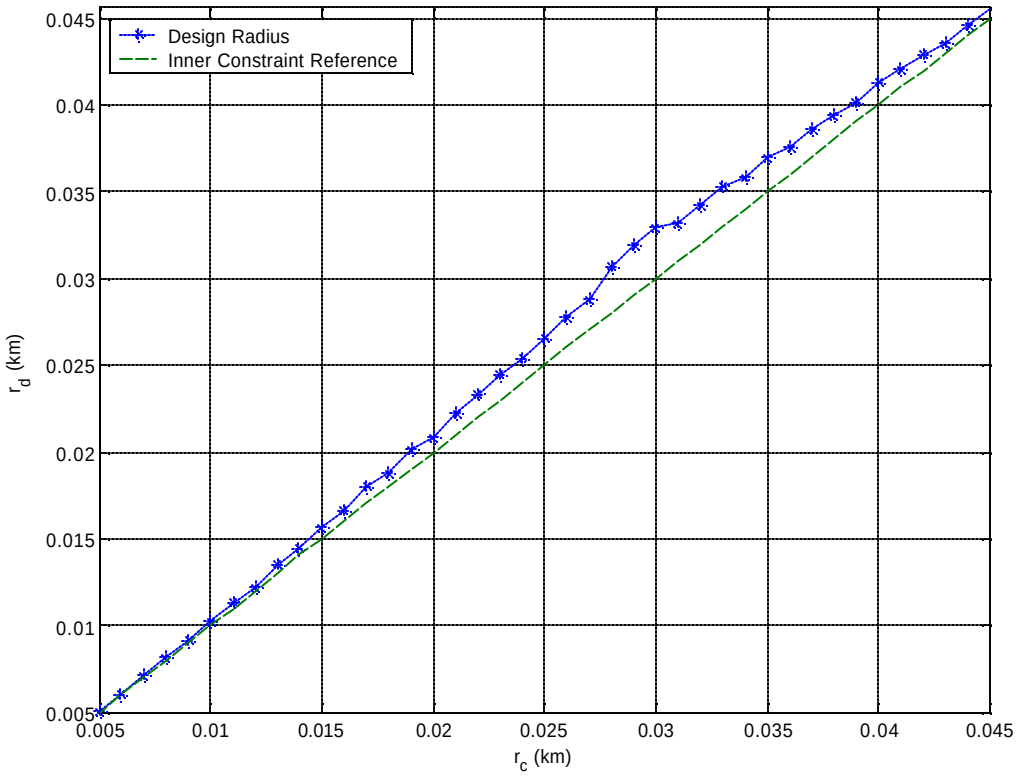


Figure 36. Design Radius Versus Inner Constraint Radius. r_0 = 50 m, Θ_y=60°, Θ_z=30°, γ_o=45°, TOF = 0.1

The dashed straight line is where the design radius would be equal to the inner constraint radius, and the starred line represents the computed design radius for each inner constraint radius. The design radii approach the inner constraint radii as the inner constraint radius shrinks with the greatest difference at the 30 m point. The greatest

difference is approximately 3 m. There does not appear to be a direct correlation between the differences in the design radii and the inner constraint radii and the total impulse shown in Figure 35. The roughness of the design radius line is a function of the increment size of the individual design radii used in the algorithm; the smaller the increment size the smoother the line.

Initial Conditions Considerations.

The analytical design method can be easily generalized to the problem where the initial position does not necessarily lie on the 'nominal' path. This can be accomplished by replacingthe nominal radius with a 'pseudo-nominal' radius in the algorithm, and placing the inner constraint radius at the minimum allowable distance to the chief. For this problem, the end point placement would need to be taken into account. If the distance from the initial point to the inner constraint radius (i.e. maximum deviation radius) is large (greater than 0.2 r_0), then the position of the end point should be added to the algorithm as shown above.

In general, the initial velocity of the deputy with respect to the chief will not be necessarily zero. The plane of the constraint torus may be chosen (if a free parameter) to better the deputy's initial relative velocity vector. The magnitude and direction of the initial velocity can have a significant affect on the Δv_t by directly adding or subtracting to the results presented.

VI. Conclusions

Two design approaches have been developed to allow a satellite to circumnavigate a chief satellite or Resident Space Object via impulsive thrusting. The first method located burn points about a nominal path in equal angles and equal times. When considering the flight path constraints, there exists a specific minimum number of burn points for which the equal-angle/equal-time flight path will be feasible. This minimum number of burn points yields the minimum total fuel expenditure for this method. This method is useful due to its inherent simplicity, and scales to the continuous case as the number of burn points goes to infinity.

The second circumnavigation analytic approach was to choose a design radius and locate the burns along a circle of that radius, with the trajectory between the burns intersecting a circle of minimum approach distance at a tangent point. This method always yielded lower fuel expenditure than the equal-angle/equal-time (EAET) method, and increased the savings over the EAET method as the total circumnavigation time of flight decreased. The analytical design method produced a relatively simple algorithm that could be used for on-board autonomous operations.

The two described methods were employed as initial guesses for a numerical optimization routine to find a minimum-fuel solution. In most cases, the local minimum was fairly close to the analytical design method in both trajectory and fuel expenditure. Thus, the analytical design approach generally always provides the best initial guess toward a minimum-fuel solution.

Appendix A. Special Case Output Data with Optimization Check

Special Case Data Output :

Optimized Total Delta Vee: 2.48949127357e-003
Initial Total Delta Vee: 2.60817455142e-003
Inputs:
thetayd = 90 deg
thetayzd= 0 deg
gamma0d= 45 deg
b= 5
ro_m= 50 km
a= 6778 km
Tot_TOF= 0.100
r_dev = 0.0100 km
TolFun = 1.0e-009
TolCon = 1.0e-009
TolX = 1.0e-009
DiffMaxChange = 1.0e-001
DiffMinChange = 1.0e-009

Gradient:	Eigenvalues of Hessian:
0.00007461	0.00038337
0.00029368	0.01170769
0.00030405	0.06065438
0.00035531	0.12869209
-0.00006748	0.74177054
-0.00017411	0.91848107
-0.00018238	1.03272863
-0.00007135	2.00804976

Norm of Gradient: 0.00061946

Initial Guess State Vector:	Optimized State Vector:
1.25664	1.43005
1.25664	1.15891
1.25664	1.14946
1.25664	1.14303
0.20000	0.25428
0.20000	0.19841
0.20000	0.18950
0.20000	0.18143

X	g(u)	Violation Mat	F(X')	F(X')-F(X)	Ck Sum
1	+1.0e-009	0 0 0 0 0 0 0	2.4894913e-003	+7.4724515e-014	0
2	+1.0e-009	0 0 0 0 0 0 0	2.4894913e-003	+2.9334521e-013	0
3	+1.0e-009	0 0 0 0 0 0 0	2.4894913e-003	+3.0243429e-013	0
4	+1.0e-009	0 0 0 0 0 0 0	2.4894913e-003	+3.0654038e-013	0
5	+1.0e-009	0 0 0 0 0 0 0	2.4894913e-003	-7.9358395e-014	0
6	+1.0e-009	0 0 0 0 0 0 0	2.4894913e-003	-1.9296023e-013	0
7	+1.0e-009	0 0 0 0 0 0 0	2.4894913e-003	-1.9792024e-013	0
8	+1.0e-009	0 0 0 0 0 0 0	2.4894913e-003	-1.7930579e-013	0
1	-1.0e-009	0 0 0 0 0 0 0	2.4894913e-003	-7.4724515e-014	0
2	-1.0e-009	0 0 0 0 0 0 0	2.4894913e-003	-2.9334521e-013	0
3	-1.0e-009	0 0 0 0 0 0 0	2.4894913e-003	-3.0243473e-013	0
4	-1.0e-009	0 0 0 0 0 0 0	2.4894913e-003	-3.0654082e-013	0
5	-1.0e-009	0 0 0 0 0 0 0	2.4894913e-003	+7.9305052e-014	0
6	-1.0e-009	0 0 0 0 0 0 0	2.4894913e-003	+1.9295633e-013	0
7	-1.0e-009	0 0 0 0 0 0 0	2.4894913e-003	+1.9785952e-013	0
8	-1.0e-009	0 0 0 0 0 0 0	2.4894913e-003	+1.7924507e-013	0
**********					
1	+1.0e-008	0 0 0 0 0 0 0	2.4894913e-003	+7.4724732e-013	0
2	+1.0e-008	0 0 0 0 0 0 0	2.4894913e-003	+2.9334547e-012	0
3	+1.0e-008	0 0 0 0 0 0 0	2.4894913e-003	+3.0243473e-012	0
4	+1.0e-008	0 0 0 0 0 0 0	2.4894913e-003	+3.0654051e-012	0
5	+1.0e-008	0 0 0 0 0 0 0	2.4894913e-003	-7.9335540e-013	0
6	+1.0e-008	0 0 0 0 0 0 0	2.4894913e-003	-1.9296049e-012	0
7	+1.0e-008	0 0 0 0 0 0 0	2.4894913e-003	-1.9789764e-012	0
8	+1.0e-008	0 0 0 0 0 0 0	2.4894913e-003	-1.7926841e-012	0
1	-1.0e-008	0 0 0 0 0 0 0	2.4894913e-003	-7.4724471e-013	0
2	-1.0e-008	0 0 0 0 0 0 0	2.4894913e-003	-2.9334525e-012	0
3	-1.0e-008	0 0 0 0 0 0 0	2.4894913e-003	-3.0243455e-012	0
4	-1.0e-008	0 0 0 0 0 0 0	2.4894913e-003	-3.0654038e-012	0
5	-1.0e-008	0 0 0 0 0 0 0	2.4894913e-003	+7.9335019e-013	0
6	-1.0e-008	0 0 0 0 0 0 0	2.4894913e-003	+1.9296019e-012	0
7	-1.0e-008	0 0 0 0 0 0 0	2.4894913e-003	+1.9789465e-012	0
8	-1.0e-008	0 0 0 0 0 0 0	2.4894913e-003	+1.7926563e-012	0

X	g(u)	Violation Mat	F(X')	F(X')-F(X)	Ck Sum
1	+1.0e-007	0 0 0 0 0 0 1	2.4894913e-003	+7.4724597e-012	1
2	+1.0e-007	0 0 0 0 0 0 0	2.4894913e-003	+2.9334540e-011	0
3	+1.0e-007	0 0 0 0 0 0 0	2.4894913e-003	+3.0243468e-011	0
4	+1.0e-007	0 0 0 0 0 0 0	2.4894913e-003	+3.0654051e-011	0
5	+1.0e-007	0 0 0 0 0 0 0	2.4894913e-003	-7.9332938e-012	0
6	+1.0e-007	0 0 0 0 0 0 0	2.4894913e-003	-1.9295616e-011	0
7	+1.0e-007	0 0 0 0 0 0 0	2.4894913e-003	-1.9789163e-011	0
8	+1.0e-007	0 0 0 0 0 0 0	2.4894913e-003	-1.7926180e-011	0
1	-1.0e-007	0 0 0 0 0 0 0	2.4894913e-003	-7.4724571e-012	0
2	-1.0e-007	0 0 0 0 0 0 0	2.4894912e-003	-2.9334531e-011	0
3	-1.0e-007	0 0 0 0 0 0 0	2.4894912e-003	-3.0243460e-011	0
4	-1.0e-007	0 0 0 0 0 0 0	2.4894912e-003	-3.0654041e-011	0
5	-1.0e-007	0 0 0 0 0 0 0	2.4894913e-003	+7.9339729e-012	0
6	-1.0e-007	0 0 0 0 0 0 0	2.4894913e-003	+1.9296525e-011	0
7	-1.0e-007	0 0 0 0 0 0 0	2.4894913e-003	+1.9790109e-011	0
8	-1.0e-007	0 0 0 0 0 0 0	2.4894913e-003	+1.7927353e-011	0
**********					
1	+1.0e-006	0 0 0 0 0 0 1	2.4894913e-003	+7.4724661e-011	1
2	+1.0e-006	0 0 0 0 0 0 0	2.4894916e-003	+2.9334572e-010	0
3	+1.0e-006	0 0 0 0 0 0 0	2.4894916e-003	+3.0243504e-010	0
4	+1.0e-006	0 0 0 0 0 0 0	2.4894916e-003	+3.0654096e-010	0
5	+1.0e-006	0 0 0 0 0 0 1	2.4894912e-003	-7.9301980e-011	1
6	+1.0e-006	0 0 0 0 0 0 1	2.4894911e-003	-1.9291631e-010	1
7	+1.0e-006	0 0 0 0 0 0 1	2.4894911e-003	-1.9784854e-010	1
8	+1.0e-006	0 0 0 0 0 0 1	2.4894911e-003	-1.7920718e-010	1
1	-1.0e-006	0 0 0 0 0 0 1	2.4894912e-003	-7.4724503e-011	1
2	-1.0e-006	0 0 0 0 0 0 1	2.4894910e-003	-2.9334499e-010	1
3	-1.0e-006	0 0 0 0 0 0 1	2.4894910e-003	-3.0243424e-010	1
4	-1.0e-006	0 0 0 0 0 0 1	2.4894910e-003	-3.0653996e-010	1
5	-1.0e-006	0 0 0 0 0 0 1	2.4894914e-003	+7.9370827e-011	1
6	-1.0e-006	0 0 0 0 0 0 0	2.4894915e-003	+1.9300532e-010	0
7	-1.0e-006	0 0 0 0 0 0 0	2.4894915e-003	+1.9794419e-010	0
8	-1.0e-006	0 0 0 0 0 0 0	2.4894915e-003	+1.7932833e-010	0

X	g(u)	Violation Mat	F(X')	F(X')-F(X)	Ck Sum
1	+1.0e-005	0 0 0 0 0 0 1	2.4894920e-003	+7.4725363e-010	1
2	+1.0e-005	0 0 0 0 0 0 0	2.4894942e-003	+2.9334902e-009	0
3	+1.0e-005	0 0 0 0 0 0 0	2.4894943e-003	+3.0243862e-009	0
4	+1.0e-005	0 0 0 0 0 0 0	2.4894943e-003	+3.0654547e-009	0
5	+1.0e-005	0 0 0 0 0 0 1	2.4894905e-003	-7.8992135e-010	1
6	+1.0e-005	0 0 0 0 0 0 1	2.4894893e-003	-1.9251596e-009	1
7	+1.0e-005	0 0 0 0 0 0 1	2.4894893e-003	-1.9741791e-009	1
8	+1.0e-005	0 0 0 0 0 0 1	2.4894895e-003	-1.7866183e-009	1
1	-1.0e-005	0 0 0 0 0 0 1	2.4894905e-003	-7.4723802e-010	1
2	-1.0e-005	0 0 0 0 0 0 1	2.4894883e-003	-2.9334169e-009	1
3	-1.0e-005	0 0 0 0 0 0 1	2.4894882e-003	-3.0243066e-009	1
4	-1.0e-005	0 0 0 0 0 0 1	2.4894882e-003	-3.0653544e-009	1
5	-1.0e-005	0 0 0 0 0 0 1	2.4894921e-003	+7.9680664e-010	1
6	-1.0e-005	0 0 0 0 0 0 0	2.4894932e-003	+1.9340565e-009	0
7	-1.0e-005	0 0 0 0 0 0 0	2.4894933e-003	+1.9837483e-009	0
8	-1.0e-005	0 0 0 0 0 0 0	2.4894931e-003	+1.7987368e-009	0
**********					
1	+1.0e-004	0 0 0 0 0 0 1	2.4894987e-003	+7.4732388e-009	1
2	+1.0e-004	0 0 0 0 0 0 0	2.4895206e-003	+2.9338204e-008	0
3	+1.0e-004	0 0 0 0 0 0 0	2.4895215e-003	+3.0247445e-008	0
4	+1.0e-004	0 0 0 0 0 0 0	2.4895219e-003	+3.0659060e-008	0
5	+1.0e-004	0 0 0 0 0 0 1	2.4894837e-003	-7.5893082e-009	1
6	+1.0e-004	0 0 0 0 0 0 1	2.4894724e-003	-1.8851248e-008	1
7	+1.0e-004	0 0 0 0 0 0 1	2.4894720e-003	-1.9311213e-008	1
8	+1.0e-004	0 0 0 0 0 0 1	2.4894740e-003	-1.7320955e-008	1
1	-1.0e-004	0 0 0 0 0 0 1	2.4894838e-003	-7.4716777e-009	1
2	-1.0e-004	0 0 0 0 0 0 1	2.4894619e-003	-2.9330866e-008	1
3	-1.0e-004	0 0 0 0 0 0 1	2.4894610e-003	-3.0239482e-008	1
4	-1.0e-004	0 0 0 0 0 0 1	2.4894606e-003	-3.0649031e-008	1
5	-1.0e-004	0 0 0 0 0 0 1	2.4894996e-003	+8.2778406e-009	1
6	-1.0e-004	0 0 0 0 0 0 0	2.4895110e-003	+1.9740938e-008	0
7	-1.0e-004	0 0 0 0 0 0 0	2.4895115e-003	+2.0268143e-008	0
8	-1.0e-004	0 0 0 0 0 0 0	2.4895098e-003	+1.8532805e-008	0

Appendix B. MATLAB* Code for Analytical Design Method†

Main Program: Analytical Design and Optimization

```
% Analytical Design Algorithm for Fast Satellite Circumnavigation
% Main Program
% Capt Stan Straight, USAF
% Air Force Institute of Technology
% 18 Feb 04

clc;clear;

global thetay thetaz gamma0 ro_m a Tot_TOF devr rd0 epsi0

%%%%%%%%%%%%%%%%%%
%       Inputs
%%%%%%%%%%%%%%%%%%
% Rotation of the path from the y-z plane about the y axis
thetayd = 60; % degrees
thetay = thetayd.*pi./180; % radians

% Rotation of the path about the z-axis
thetazd = 30; % degrees
thetaz = thetazd*pi/180; % radians

% Define the initial angle of the initial position from the y-axis
% direction
gamma0d = 45; % degrees
gamma0 = gamma0d*pi/180; % radians

% Input the radius of the reference orbit in kilometers
a = 6778; % km

% Input the time of flight for the complete circumnavigation as a fraction
% of the reference orbit's period
Tot_TOF = .1;

% Define the nominal radius
ro_m = 100; % m
r0 = ro_m./1000; % km

% Maximum Deviation Radius
devr = 0.04; %km

% Minimum Radius to search over (fraction of the devr)
minrc = 1.001;

% Establish the initial radius from the nominal path and the angle from the
```

* Version 6.5, Release 13. See Reference 6.
† Only core programs and sub-programs shown: main program and sub-programs call other sub-programs for formatting data for plotting not included

```
% minor axis
rd0 = 0;
epsi0 = 0;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       End Inputs
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Inner constraint radius
rc = r0 - devr;
maxrc = rc+devr;

% Outside constraint radius
rt = r0 + devr;

% Calculate alpha angle for algorithm
al = acos(rc./r0);

% Establish the search region
rd = minrc.*rc:.0001:maxrc;

% Call the algorithm as a function firstorddes
[states,tot_delvee,const,numbpts] = firstorddes(rt,rc,rd);

% Check for the plot of the points where there is a change in the number of
%  burn points between each different case
combo = [rd' numbpts'];
jun = 1;
for s = 1:length(rd)-1
   if combo(s,2) > combo(s+1,2) | combo(s,2) < combo(s+1,2);
      ptbpt(jun,:) = combo(s,:);
      jun = jun +1;
   end
end

% Determine the minimum point, and the design radius producing the minimum
% value
[yuck,ind] = min(tot_delvee);
rdyuck = rd(ind);

% Add one to the number of burn points vector to determine the acctual
% number of burn points within a given state
bvec = numbpts + 1;

% Find the plots for the min state
smin = states(:,ind);
for g = 1:length(smin)-3
   if sum([smin(g) smin(g+1)]) > 0
      xstmin(g) = smin(g);
   end
end

% Place the burn points for the minimum design vector
[bposi,rnom] = plCostfcnav1(xstmin);

% Use the optimization program to evaluate a new minima if available
```

```
[xstop,fvalop] = genoptimization(xstmin);
% Place the burn points for the optimized state vector
[bposiop,rnomo] = plCostfcnav1(xstop);

% Create the actual intermediate points for each state vector
cvecmin = plotintptsav1(xstmin);
cvecop = plotintptsav1(xstop);

% Plot the total impulse as a function of the design radii
figure(1)
plot(rd,tot_delvee,'.-')
%title(['Analytical Approach: Total Delta Vee vs Change in Design Radius, devr=',num2str(devr)])
xlabel('r_d (km)')
ylabel('\Delta v_t (km/s)')
hold on
for h =1:length(ptbpt)
plot([ptbpt(h,1) ptbpt(h,1)],[min(tot_delvee) max(tot_delvee)],'r:')
end
hold off

% Plot the number of burn points
figure(2)
 plot(rd,bvec)
 %title('Number of Burn Points Computed vs. Design Radius')
xlabel('r_d (km)')
 ylabel('Number of Burn Points (Includes the Initial Point)')

 % Call a function to create the torus surface representing the maximum
 % deviation constraint
 [rdx,rdy,rdz] = torusmaker1(r0,devr,thetay,thetaz);

figure(3)
plot3(bposi(:,1),bposi(:,2),bposi(:,3),'ro',cvecmin(:,1),cvecmin(:,2),cvecmin(:,3),'r-',...
   bposiop(:,1),bposiop(:,2),bposiop(:,3),'k+',cvecop(:,1),cvecop(:,2),cvecop(:,3),'k--',...
   rnom(:,1),rnom(:,2),rnom(:,3),'-.')
grid on
% title(['Optimized Flight Path, Thetay=',num2str(thetayd),', Thetaz=',num2str(thetazd),...
%     ', Gamma0',num2str(gamma0d),', b=',num2str(numbpts(ind)+1),', TOF=',num2str(Tot_TOF)])
legend('Design Burn Points','Design Path','Optimized Burn Points','Optimized Path','Nominal Path')
xlabel('x (km)')
ylabel('y (km)')
zlabel('z (km)')
axis square
hold on
mesh(rdx,rdy,rdz)
hidden off
shading flat
colormap([0.9 0.9 0.9])
hold off
axis equal

% Create the plot showing the deviation from the nominal path for each the
% design state and the optimized state vector
[cmi,tvecmi] = planonlincont(xstmin);
```

```
[co,tveco] = planonlincont(xstop);
 cmin = zeros(length(tvecmi)+1,1)';
 tvecmin = zeros(length(tvecmi)+1,1)';
 cmin(2:length(tvecmi)+1) = cmi;
 tvecmin(2:length(tvecmi)+1)= tvecmi;

 cop = zeros(length(tvecmi)+1,1)';
 tvecop = zeros(length(tvecmi)+1,1)';
 cop(2:length(tvecmi)+1) = co;
 tvecop(2:length(tvecmi)+1)= tveco;
 % Compute a line for visualizing the constraint
dep = ones(length(tvecmi)+1,1)';
devrplot = dep.*devr;

figure(4)
plot(tvecmin,cmin,'.-',tvecop,cop,'+-',tvecmin,devrplot,'--')
% title(['Deviation of Actual Path','Thetay=',num2str(thetayd),', Thetaz=',num2str(thetazd),...
%     ', Gamma0',num2str(gamma0d),', b=',num2str(numbpts(ind)+1),', TOF=',num2str(Tot_TOF)])
xlabel('Time (fraction of TOF)')
ylabel('\rho (km)')
grid on
axis([0 1 0 devr+devr.*.05])
legend('Design Path','Optimized Path','Constraint Boundary',4)

   b0= (length(xstmin)+2)./4;

   % Create a dummy variable to populate with the EAET method
    xst0(1:2.*b0) = zeros(2.*b0,1)';

   % Equal Angle/Equal Time
       gammai = 1./b0;
       ti = (1./b0);
    for i = 1:b0-1
       xst0(2.*b0 + i) =gammai;
       xst0(2.*b0 + i+(b0-1)) = ti;
   end

   % Compute the total impulse for the EAET
   eqtot_delvee = Costfcnav1(xst0);

disp(['Equal Burn/Equal Time Total De lta Vee: ',num2str(eqtot_delvee)])
disp(['Design Total Delta Vee:',num2str(yuck)])
disp(['        Found at an rd of:',num2str(rdyuck)])
disp(['Optimized Total Delta Vee:',num2str(fvalop)])
   % Compute the percentage difference between the Design and Optimized
   % Total Delta Vee
   designper = ((eqtot_delvee - yuck)./eqtot_delvee).*100;
   optimper = ((eqtot_delvee - fvalop)./eqtot_delvee).*100;
disp('  ')
disp(['Percentage Saved by Design:',num2str(designper),'%'])
disp(['Percentage Saved by Optimization:',num2str(optimper),'%'])
if const(ind) > 0
   disp('*******Designed State Exceeds Constraints*******')
end
```

Subprogram: Analytical Design Function

```
function [states,tot_delvee,const,numbpts] = firstorddes(rt,rc,rd)
% This function is the analytical design method given the inner
% contraint radius, rc, the last point radius rt, and the search region for
% the design radius (rd).  The design radius must be larger than one point.
%  The output is the states for each design radius, the total impulse for
%  those states and the constraint matrix for those states.  If any number
%  in the constraint matrix is greater than 0, then the state exceeds the
%  constraints.  The format for calling this function is
%  [states,tot_delvee,const] = firstorddes(rt,rc,rd)

global thetay thetaz gamma0 ro_m a Tot_TOF devr rd0 epsi0

r0 = ro_m./1000; % km
maxrc = rc+devr;
al = acos(rc./r0);
sigma = 0;

for k = 1:length(rd)
   % First path length
   p(1,k) = sqrt(r0^2-rc^2)+sqrt(rd(k)^2-rc^2);
    pf(k) = sqrt(rt^2-rc^2)+sqrt(rd(k)^2-rc^2);

   gamma(1,k) = asin((p(1,k).*rc)./(r0.*rd(k)));
      gammaf(k) = asin((pf(k).*rc)./(rt.*rd(k)));

   counter = 1;

   while sigma(k) < (2.*pi-gammaf(k))
      gamma(counter+1,k) = 2.*acos(rc./rd(k));
      counter = counter + 1;
      sigmat = sum(gamma,1);
      sigma(k) = sigmat(k);
   end

   if sigma(k) > (2.*pi - gammaf(k))
      gamt(1:counter-1,k) = gamma(1:counter-1,k);
      sigmaft = sum(gamt,1);
      sigmaf(k) = sigmaft(k);
      gamma(counter,k) = 2.*pi - sigmaf(k) - gammaf(k);
   end

   sigma(k+1) = 0;
end

% Normalize the gamms to 2 pi
gamman = gamma./(2.*pi);

% Set the time between the burn points to the same ratio with respect to
% the total time
timen = gamman;
% Compute the normalized distance from the nominal
rdn = abs(r0-rd)./devr;
```

```
% Set all the points except for the last one on the inside of the nominal
% within the nominal's plane; place the last point on the outside of the
% torus in the nominal's plane
epsi(1:length(rd)-1) = 0.5;
epsi(length(rd)) = 0;

% Build each unique state vector
gut = size(gamman);

for h = 1:length(rd)
   for q = 1:gut(1)
      if gamman(q,h)>0
         numbpts(h) = q;
         bi = numbpts;
      end
   end
end
% Add one to the number of burn points vector to determine the acctual
% number of burn points within a given state
bvec = numbpts + 1;
for u = 1:length(rd)
   xstn(1:bi(u)) = rdn(u);
   xstn(bi(u)+1) = abs(r0-rt)./devr;
   xstn(bi(u)+2:2.*bi(u)+1) = 0.5;
   if rt >= r0
   xstn(2.*bi(u)+2) = 0;
   else
   xstn(2.*bi(u)+2)=0.5;
   end
   xstn(2.*bi(u)+3:3.*bi(u)+2) = gamman(1:numbpts(u),u);
   xstn(3.*bi(u)+3:4.*bi(u)+2) = timen(1:numbpts(u),u);
   if u == 1
      states = zeros(length(xstn),length(rd));
   end
   xstn = xstn';
   states(1:length(xstn),u) = xstn;
   % Compute the cost function value for the new state vector
   tot_delvee(u) = Costfcnav1(xstn);
      % Compute the constraint function for the state to determine if the
   % constraints are violated
   [dum,ceq] = anonlincon(xstn);
   c(u,1:length(dum)) = dum;
   const(u)=0;
   constc(u) = 0;
   for y = 1:length(c)
      if c(u,y) > 0
         const(u) = const(u)+1;
         constc(u) = 1;
      end
   end
   xstn = 0;
   dum = [];
end
```

Sub-Subprogram: Cost Function Evaluation – Analytical Design

```
function tot_delvee = costfcnav1(xst)
% This is the cost function to be used with fmincon.  It computes the total
% delta vee required to perform a circumnavigation, global variables are used and
% should be defined by the main program, additionally the state vector xst should be
% of the format rd's, epsilon's, gamma's then time and of 4*b -2 length.

global thetay thetaz gamma0 ro_m a Tot_TOF devr rd0 epsi0

% Calculate the mean motion of the reference orbit in rad/s
n = sqrt((3.98601*10^5)/a^3);

% Compute the time of circumnavigation
refperiod = 2*pi/n;  % seconds
tot_toc = Tot_TOF*refperiod; % seconds

b= (length(xst)+2)./4;
bmax = b;
% Path radius in kilometers
r0 = ro_m/1000; % kilometers

% Calculate the real values from the normalized states
rds = xst(1:b).*devr;
epsis = xst(b+1:2.*b).*2.*pi;
gamms = xst(2.*b+1:3.*b-1).*2.*pi;
tims = xst(3.*b:4.*b-2).*tot_toc;

% Define the initial position vector of the initial burn point
      rdx0 = rd0.*sin(epsi0).*cos(thetay).*cos(thetaz) +
(r0+rd0.*cos(epsi0)).*sin(gamma0).*cos(thetaz).*sin(thetay)...
         -(r0+rd0.*cos(epsi0)).*cos(gamma0).*sin(thetaz);
      rdy0 = (r0+rd0.*cos(epsi0)).*cos(gamma0).*cos(thetaz)+rd0.*sin(epsi0).*cos(thetay).*sin(thetaz)...
         +(r0+rd0.*cos(epsi0)).*sin(gamma0).*sin(thetay).*sin(thetaz);
      rdz0 = (r0+rd0.*cos(epsi0)).*sin(gamma0).*cos(thetay)-rd0.*sin(epsi0).*sin(thetay);

rint = [rdx0;rdy0;rdz0]';

% Define the initial velocity vector of the initial burn point
vint = [0 0 0];

% Establish the initial velocity of the interceptor at the initial point
dvf(1,:) = vint;

for s = 1:b
      if s < b
         rd = rds(s);
         epsi = epsis(s);
         toc = tims(s);
         if s ==1
         gamma_j = gamms(s)+gamma0;
         else
         gamma_j = gamms(s)+gamma_j;
         end
```

```
    elseif s == b
        rd = rds(s);
        epsi = epsis(s);
        gamma_j = (2.*pi - sum(gamms))+gamma_j;
        toc = tot_toc - sum(tims);
    end
    %Position of the next burn point
    rdx = rd.*sin(epsi).*cos(thetay).*cos(thetaz) +
(r0+rd.*cos(epsi)).*sin(gamma_j).*cos(thetaz).*sin(thetay)...
        -(r0+rd.*cos(epsi)).*cos(gamma_j).*sin(thetaz);
    rdy = (r0+rd.*cos(epsi)).*cos(gamma_j).*cos(thetaz)+rd.*sin(epsi).*cos(thetay).*sin(thetaz)...
        +(r0+rd.*cos(epsi)).*sin(gamma_j).*sin(thetay).*sin(thetaz);
    rdz = (r0+rd.*cos(epsi)).*sin(gamma_j).*cos(thetay)-rd.*sin(epsi).*sin(thetay);

    rfin = [rdx;rdy;rdz]';
    % Compute the velocity needed to go from the current point to the next point
    % in the given time of flight
    dvi(s,:) = hillsvel2(rint,rfin,toc,n);
    if s < b

     % Compute the actual velocity at the next point
        dvf(s+1,:) = hillsvelf(rint,dvi(s,:),toc,n);
    end
    % Reset the position of the next burn point to the current burn
    % point to propagate the next sequential burn point along the path
    rint = rfin;

end

% Compute the change in velocity
delvee_vec = dvf - dvi;

if b == 1
   delvee_mag_burn(b) = norm(delvee_vec(b,:));
end

for g = 1:b
delvee_mag_burn(g) = norm(delvee_vec(g,:));
end

tot_delvee = sum(delvee_mag_burn);
```

Sub-Subprogram: Compute Impulse Between Points Using Hill's Equations

```
function vint = hillsvel2(rint,rfin,tof,n)
% This program will take an initial and final position row vectors defined
% in the LVLH frame, the time of flight between the points and output the
% initial velocity required to achieve these parameters.  The output is in
% a row vector format for the velocity components in the LVLH frame.  The
% time of flight needs to be in seconds and the value of n has to be in
% rad/s.  n is the mean motion of the reference orbit.
```

```
% Define the value of psi
ang = n * tof;

% Input the phi rr and phi rv matrices

 phirr = [(4-3*cos(ang)),0,0;6*(sin(ang)-ang),1,0;0,0,cos(ang)];

 inphirv = [n*(3*ang-4*sin(ang))/(-8+8*cos(ang)+3*ang*sin(ang)),-2*n*(cos(ang)-1)/(-
8+8*cos(ang)+3*ang*sin(ang)),0;
    2*n*(cos(ang)-1)/(-8+8*cos(ang)+3*ang*sin(ang)),n*sin(ang)/(8-8*cos(ang)-3*ang*sin(ang)),0;
    0,0,n/sin(ang)];

% compute the required initial velocity
vint = (inphirv*(rfin'-phirr*rint'))';
%end
```

Sub-Subprogram: Propogate Velocity from Previous Burn Point with Hill's Equations

```
function vfin = hillsvelf(rint,vint,tof,n)
% This program will take an initial position  and initial velocity row vectors defined
% in the LVLH frame, the time of flight between the points and output the
% velocity at the end of the time of flight required to achieve these parameters.
% The output is in
% a row vector format for the velocity components in the LVLH frame.  The
% time of flight needs to be in seconds and the value of n has to be in
% rad/s.  n is the mean motion of the reference orbit.

% Define the value of psi
ang = n * tof;

% Input the phi vr and phi vv matrices

 phivr = [3*n*sin(ang),0,0;6*n*(cos(ang)-1),0,0;0,0,-n*sin(ang)];

 phivv = [cos(ang),2*sin(ang),0;-2*sin(ang),-3+4*cos(ang),0;0,0,cos(ang)];
 % compute the required final velocity
vfin = (phivr*rint'+phivv*vint')';
```

Subprogram: Perform Optimization Using *fmincon*

```
% Compute the optimized path given the global constraints and initial guess
% state vector
% Capt Stan Straight

function [xst,fval] = genoptimization(xst0)

global thetay thetaz gamma0 ro_m a Tot_TOF devr rd0 epsi0
% Establish the options parameters for the optimization function
dmaxchange = 1e-1;
dminchange = 1e-9;
```

```
TolFun = 2e-9;
TolCon = 1e-9;
TolX = 1e-9;

% Calculate the mean motion of the reference orbit in rad/s
n = sqrt((3.98601*10^5)./a^3);
% Compute the time of circumnavigation
refperiod = 2*pi./n;  % seconds
tot_toc = Tot_TOF*refperiod; % seconds
% Compute the number of burn points
b= (length(xst0)+2)./4;
bmax = b;

% Path radius in kilometers
r0 = ro_m/1000; % kilometers
% Establish the A matrix contraint and the b matrix constraint
A = zeros(4.*b-2,4.*b-2);
A(1,2.*b+1:3.*b-1) = ones(1,b-1);
A(2,3.*b:4.*b-2) = ones(1,b-1);
% Establish the b matrix contraint
bineq = zeros(4.*b-2,1);
bineq(1) = 1;
bineq(2) = 1;

% Define the lower and upper bounds of the functions
for k = 1:4.*b-2
   ub(k) = 1;
end
ub=ub';
for p = 1:4.*b-2
   lb(p) = 0;
end
lb(3.*b:4.*b-2) = 1e-7;
lb = lb';
options =
optimset('LargeScale','off','MaxFunEvals',20000,'MaxIter',500,'TolFun',TolFun,'TolCon',TolCon,...
   'TolX',TolX,'Display','iter','DiffMaxChange',dmaxchange,'DiffMinChange',dminchange);

% Use fmincon function to find the state vector that produces the minimum total deltavee
[xst,fval,exitflag,output,lambda,grad,hessian] =
fmincon(@Costfcngv1,xst0,A,bineq,[],[],lb,ub,@gnonlincon,options);
```

Sub-Subprogram: Cost Function Evaluation for Optimization.

```
function tot_delvee = costfcngv1(xst)
% This is the cost function to be used with fmincon.  It computes the total
% delta vee required to perform a circumnavigation, global variables are used and
% should be defined by the main program, additionally the state vector xst should be
% of the format rd's, epsilon's, gamma's then time and of 4*b-2 length.

global thetay thetaz gamma0 ro_m a Tot_TOF devr rd0 epsi0

b= (length(xst)+2)./4;
```

```
bmax = b;

% Calculate the mean motion of the reference orbit in rad/s
n = sqrt((3.98601*10^5)/a^3);

% Compute the time of circumnavigation
refperiod = 2*pi/n;  % seconds
tot_toc = Tot_TOF*refperiod; % seconds

% Path radius in kilometers
r0 = ro_m/1000; % kilometers

% Calculate the real values from the normalized states
rds = xst(1:b).*devr;
epsis = xst(b+1:2.*b).*2.*pi;
gamms = xst(2.*b+1:3.*b-1).*2.*pi;
tims = xst(3.*b:4.*b-2).*tot_toc;

% Define the initial position vector of the initial burn point
       rdx0 = rd0.*sin(epsi0).*cos(thetay).*cos(thetaz) +
(r0+rd0.*cos(epsi0)).*sin(gamma0).*cos(thetaz).*sin(thetay)...
          -(r0+rd0.*cos(epsi0)).*cos(gamma0).*sin(thetaz);
       rdy0 = (r0+rd0.*cos(epsi0)).*cos(gamma0).*cos(thetaz)+rd0.*sin(epsi0).*cos(thetay).*sin(thetaz)...
          +(r0+rd0.*cos(epsi0)).*sin(gamma0).*sin(thetay).*sin(thetaz);
      rdz0 = (r0+rd0.*cos(epsi0)).*sin(gamma0).*cos(thetay)-rd0.*sin(epsi0).*sin(thetay);

rint = [rdx0;rdy0;rdz0]';

% Define the initial velocity vector of the initial burn point
vint = [0 0 0];

% Establish the initial velocity of the interceptor at the initial point
dvf(1,:) = vint;

for s = 1:b
      if s < b
         rd = rds(s);
          epsi = epsis(s);
          toc = tims(s);
         if s ==1
          gamma_j = gamms(s)+gamma0;
          else
          gamma_j = gamms(s)+gamma_j;
          end
      elseif s == b
          rd = rds(s);
         epsi = epsis(s);
          gamma_j = (2.*pi - sum(gamms))+gamma_j;
         toc = tot_toc - sum(tims);
      end
      %Position of the next burn point
      rdx = rd.*sin(epsi).*cos(thetay).*cos(thetaz) +
(r0+rd.*cos(epsi)).*sin(gamma_j).*cos(thetaz).*sin(thetay)...
          -(r0+rd.*cos(epsi)).*cos(gamma_j).*sin(thetaz);
```

```
    rdy = (r0+rd.*cos(epsi)).*cos(gamma_j).*cos(thetaz)+rd.*sin(epsi).*cos(thetay).*sin(thetaz)...
        +(r0+rd.*cos(epsi)).*sin(gamma_j).*sin(thetay).*sin(thetaz);
    rdz = (r0+rd.*cos(epsi)).*sin(gamma_j).*cos(thetay)-rd.*sin(epsi).*sin(thetay);

    rfin = [rdx;rdy;rdz]';

    % Compute the velocity needed to go from the current point to the next point
    % in the given time of flight
    dvi(s,:) = hillsvel2(rint,rfin,toc,n);
    if s < b
     % Compute the actual velocity at the next point
        dvf(s+1,:) = hillsvelf(rint,dvi(s,:),toc,n);
    end
    % Reset the position of the next burn point to the current burn
    % point to propagate the next sequential burn point along the path
    rint = rfin;
end

% Compute the change in velocity
delvee_vec = dvf - dvi;

if b == 1
   delvee_mag_burn(b) = norm(delvee_vec(b,:));
end

for g = 1:b
delvee_mag_burn(g) = norm(delvee_vec(g,:));
end

tot_delvee = sum(delvee_mag_burn);
```

Sub-Subprogram: Determine Flight Path Constraints

```
function [c,ceq] = gnonlincon(xst)

global thetay thetaz gamma0 ro_m a Tot_TOF devr rd0 epsi0

% Define the number of intermediate points
z = 20;
% Calculate the mean motion of the reference orbit in rad/s
n = sqrt((3.98601*10^5)/a^3);
% Compute the time of circumnavigation
refperiod = 2*pi/n;  % seconds
tot_toc = Tot_TOF*refperiod; % seconds

% Path radius in kilometers
r0 = ro_m/1000; % kilometers
b= (length(xst)+2)./4;
bmax = b;

% Calculate the real values from the normalized states
rds = xst(1:b).*devr;
epsis = xst(b+1:2.*b).*2.*pi;
```

```
gamms = xst(2.*b+1:3.*b-1).*2.*pi;
tims = xst(3.*b:4.*b-2).*tot_toc;

% Define the initial position vector of the initial burn point
      rdx0 = rd0.*sin(epsi0).*cos(thetay).*cos(thetaz) +
(r0+rd0.*cos(epsi0)).*sin(gamma0).*cos(thetaz).*sin(thetay)...
         -(r0+rd0.*cos(epsi0)).*cos(gamma0).*sin(thetaz);
      rdy0 = (r0+rd0.*cos(epsi0)).*cos(gamma0).*cos(thetaz)+rd0.*sin(epsi0).*cos(thetay).*sin(thetaz)...
         +(r0+rd0.*cos(epsi0)).*sin(gamma0).*sin(thetay).*sin(thetaz);
      rdz0 = (r0+rd0.*cos(epsi0)).*sin(gamma0).*cos(thetay)-rd0.*sin(epsi0).*sin(thetay);

rint = [rdx0;rdy0;rdz0]';

% Define the initial velocity vector of the initial burn point
vint = [0 0 0];

% Establish the initial velocity of the interceptor at the initial point
dvf(1,:) = vint;

for s = 1:b
      if s < b
         rd = rds(s);
         epsi = epsis(s);
         toc = tims(s);
         if s ==1
         gamma_j = gamms(s)+gamma0;
         else
         gamma_j = gamms(s)+gamma_j;
         end
      elseif s == b
         rd = rds(s);
         epsi = epsis(s);
         gamma_j = (2.*pi - sum(gamms))+gamma_j;
         toc = tot_toc - sum(tims);
      end

      %Position of the next burn point
      rdx = rd.*sin(epsi).*cos(thetay).*cos(thetaz) +
(r0+rd.*cos(epsi)).*sin(gamma_j).*cos(thetaz).*sin(thetay)...
         -(r0+rd.*cos(epsi)).*cos(gamma_j).*sin(thetaz);
      rdy = (r0+rd.*cos(epsi)).*cos(gamma_j).*cos(thetaz)+rd.*sin(epsi).*cos(thetay).*sin(thetaz)...
         +(r0+rd.*cos(epsi)).*sin(gamma_j).*sin(thetay).*sin(thetaz);
      rdz = (r0+rd.*cos(epsi)).*sin(gamma_j).*cos(thetay)-rd.*sin(epsi).*sin(thetay);

      rfin = [rdx;rdy;rdz]';
      % Compute the velocity needed to go from the current point to the next point
      % in the given time of flight
      dvi(s,:) = hillsvel2(rint,rfin,toc,n);

      % Compute the vector of intermediate points within the
      cirfin = inthillsoptnonlin(rint,dvi(s,:),toc,n,z);
      q(s) = (z+1+(s-2).*z)';
      cvec(q(s):q(s)+z-1,:) = cirfin;
```

```
        % Reset the position of the next burn point to the current burn
        % point to propagate the next sequential burn point along the path
        rint = rfin;

end

% Create a rotation matrix from the LVLH frame to the circumnav frame given
% the values of gamma0, thetay, and thetaz
rotxtop = [cos(thetay).*cos(thetaz),cos(thetay).*sin(thetaz),-1.*sin(thetay);...
        cos(thetaz).*sin(gamma0).*sin(thetay)-cos(gamma0).*sin(thetaz),...
        cos(gamma0).*cos(thetaz)+sin(thetay).*sin(thetaz).*sin(gamma0),cos(thetay).*sin(gamma0);...
        cos(gamma0).*cos(thetaz).*sin(thetay)+sin(gamma0).*sin(thetaz),...
        -1.*(cos(thetaz).*sin(gamma0))+cos(gamma0).*sin(thetay).*sin(thetaz),...
        cos(gamma0).*cos(thetay)];

pathvec = zeros(length(cvec),3);

% Create the c matrix by determining the distance of the intermediate
% position to the position of the nominal path.
for m = 1:length(cvec)
    % Extract each intermediate position
    cvecp = cvec(m,:)';
     % Rotate each vector into the pqw, then take the projection onto the
    % pq frame
     cvecprot(m,:) = (rotxtop*cvecp)';
    cvecproj(m,:) = cvecprot(m,2:3);
    pathvec(m,2:3) = r0.*(cvecproj(m,:)./norm(cvecproj(m,:)));
end

cmat = cvecprot-pathvec;

 for w = 1:length(cvec)
    c(w) = norm(cmat(w,:))-devr;
 end

% By setting this quantity to zero it eliminates the nonlinear equality constraint
% requirement
ceq = 0;
```

Bibliography

1. Arora, J. S. *Introduction to Optimum Design*. Boston: McGraw-Hill, 1989.

2. Gill, P. E., W. Murray, and M. H. Wright. *Practical Optimization*. San Diego: Academic Press, 1981.

3. Gim, D.W. and K. T. Alfriend, "The State Transition Matrix of Relative Motion For the Perturbed Non-Circular Reference Orbit", *J. of Guidance, Control, and Dynamics*, Vol. 26, No. 6, 2003, pp. 956-971.

4. Howard, R. and T. Storch, "Δv Requirements for Staring and Expedient Circular Circumnavigations", AIAA 2001-4740.

5. Lovell, T. A. and S. G. Tragesser, "Analysis of the Reconfiguration and Maintenance of Close Spacecraft Formations", *AAS/AIAA Space Flight Mechanics Meeting*, Ponce, Puerto Rico. 9-13 Feb. 2003. Paper No. AAS 03-139.

6. *MATLAB: Optimization Toolbox*. Version 2.2. Computer software. Copyright 1984-2002. The MathWorks, Inc.

7. Meissinger, H. F. and J. T. Collins, "Mission Design and System Requirements for a Multiple-Function Orbital Transfer Vehicle", *AIAA Space Technology Conference and Exposition*, AIAA Paper 1999-4439, Albuquerque, New Mexico. 28-30 September 1999. *http://www.smad.com/analysis/OTVpaper.pdf*

8. Moore, L. C. and D. A. Smith, Duke Connected Curriculum Project, Copyright 1997-2003. *http://www.math.duke.edu/education/ccp/materials/mvcalc/parasurfs/para3.html*

9. "Optimization Toolbox: For Use with MATLAB, User's Guide, Version 2", The Math Works, Inc., Online Only. Revised for Version 2.2 Release 13. July 2002.

10. Partch, Russell E., Vern Baker, and Clark Keith. "Autonomous Proximity Satellites." *Air Force Research Laboratory, Technology Horizons: Research for America's Future*, 4: 16-18 (December 2003).

11. Sabol, C., R. Burns, and C. A. Mclaughlin, "Satellite Formation Flying Design and Evolution", *Journal of Spacecraft and Rockets*, Vol. 38, No. 2, March-April 2001, pp. 270-278.

12. Schweighart, S. and R. Sedwick, “A Perturbative Analysis of Geopotential Disturbances for Satellite Cluster Formation Flying”, *2001 IEEE Aerospace Conference*, Big Sky, MT, Mar. 10-17, 2001, Piscataway, NJ, IEEE, 2001, p. 1001 – 1019.

13. Vadali, S. R., H. Schaub, and K. T. Alfriend, “Initial Conditions and Fuel-Optimal Control for Formation Flying of Satellites”, *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 1999.

14. Vallado, D. A., *Fundamentals of Astrodynamics and Applications, published as part of the Space Technology Series* by The McGraw-Hill Companies, Inc., College Custom Series. 1997.

15. Wiesel, W. E., “The Dynamics of Relative Satellite Motion”, *Advances in the Astronautical Sciences, Spaceflight Mechanics 2001*. 108:869-880.

16. Wiesel, W. E., *Spaceflight Dynamics, Second Edition*, Boston: Irwing/McGraw-Hill, 1997.

Vita

Captain Stanley D. Straight graduated from F. J. Reitz High School in Evansville, Indiana. He entered into undergraduate studies at the University of Colorado-Boulder where he graduated with a Bachelor of Sciences degree in Aerospace Engineering Sciences in May 1998. He was commissioned through the Detachment 105 AFROTC at the University of Colorado-Boulder in June 1998.

His first assignment was to Undergraduate Space and Missile Training at Vandenberg AFB, California in July 1998. Upon completion of Spacelift Initial Qualification Training, he was assigned to the 45th Range Squadron, Cape Canaveral AFS, Florida where he served as a Range Operations Commander, Range Control Officer and Instructor. In 2001, he was chosen to be the Executive Officer for the 45th Logistics Group, Cape Canaveral AFS, Florida. In August 2002, he entered the Graduate School of Engineering and Management, Air Force Institute of Technology. Upon graduation, he will be assigned to the Space Vehicles Directorate, Air Force Research Laboratory, Kirtland AFB, New Mexico.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From – To)*
12-03-2004	**Master's Thesis**	Jun 2003 – Mar 2004

4. TITLE AND SUBTITLE

MANEUVER DESIGN FOR FAST SATELLITE CIRCUMNAVIGATION

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Straight, Stanley D., Captain, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way, Building 641
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GA/ENY/04-M05

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFRL/VSES
Attn: Dr. T. Alan Lovell
3550 Aberdeen Ave, SE DSN: 263-4132
Kirtland AFB, NM 87117 e-mail: Thomas.Lovell@kirtland.af.mil

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The feasibility of satellite operations in close proximity to a reference satellite is of interest for both civilian and military applications. One such operation is circular circumnavigation in a time period less than the orbital period of the reference satellite. This thesis investigates a guidance scheme for such maneuvers involving impulsive burns at specific points within a specified toroidal region centered on the circular-orbiting reference satellite. Two analytical methods for determining the magnitude and direction of the impulses are demonstrated. These methods are then used as initial estimates in an optimization scheme to produce the minimum total required impulse.

15. SUBJECT TERMS

Spacecraft Proximity Operations, Satellite Formation Operations, Rendezvous Spacecraft, Rendezvous Trajectories, Spacecraft, Trajectories, Maneuvers, Celestial Mechanics, Orbits

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Steven G. Tragesser (ENY)
U	U	U	UU	103	19b. TELEPHONE NUMBER *(Include area code)* (937) 255-3636; e-mail: Steven.Tragesser@afit.edu